NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

ROBOTIC ARM MANIPULATION LABORATORY WITH A SIX DEGREE OF FREEDOM JACO ARM

by

Ronald H. Palacios

December 2015

Thesis Advisor:	Richard M. Harkins
Second Reader:	Peter Crooker

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704–0188*

1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** December 2015	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE ROBOTIC ARM MANIPULATION LABORATORY WITH A SIX DEGREE OF FREEDOM JACO ARM	**5. FUNDING NUMBERS**
6. AUTHOR(S) Ronald H. Palacios	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number ____N/A____.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited	**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

The JACO six degree of freedom robotic arm and associated software is characterized for use by students in a Robotic Manipulation Laboratory. The lab was implemented to help students understand tele-operation techniques with a sophisticated Kinova JACO robotic arm. The purpose of the research was to follow up on Jacinto's work and expose students to forward kinematics via Denavit-Hartenberg (DH) parameters for robotic arm manipulation in a lab environment for various experiments. Jacinto demonstrated the viability of resistive glove control via simulation through the Robot Operating System (ROS) interface. He was not able to demonstrate real-time glove control. Here, we take an intermediate approach and introduce a virtual joystick. Manipulation experiments with various effector loads and Cartesian trajectories indicate a 1.0 kg load limit in the fully extended mode and 1.5 kg limit in a midrange mode. This verifies vendor specifications for the arm and is acceptable for light-load daily mobility requirements.

14. SUBJECT TERMS robotic arm, lagrangian dynamics, kinematics, inverse kinematics, DH parameters	**15. NUMBER OF PAGES** 75
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540–01-280-5500 Standard Form 298 (Rev. 2–89) Prescribed by ANSI Std. 239–18

THIS PAGE INTENTIONALLY LEFT BLANK

ROBOTIC ARM MANIPULATION LABORATORY WITH A SIX DEGREE OF FREEDOM JACO ARM

Ronald H. Palacios
Lieutenant Junior Grade, Peruvian Navy
B.S., Escuela Naval del Peru, 2008

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN APPLIED PHYSICS

from the

NAVAL POSTGRADUATE SCHOOL
December 2015

Approved by: Richard M. Harkins
Thesis Advisor

Peter Crooker
Second Reader

Kevin Smith
Chair, Department of Physics

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The JACO six degree of freedom robotic arm and associated software is characterized for use by students in a Robotic Manipulation Laboratory. The lab was implemented to help students understand tele-operation techniques with a sophisticated Kinova JACO robotic arm. The purpose of the research was to follow up on Jacinto's work and expose students to forward kinematics via Denavit-Hartenberg (DH) parameters for robotic arm manipulation in a lab environment for various experiments. Jacinto demonstrated the viability of resistive glove control via simulation through the Robot Operating System (ROS) interface. He was not able to demonstrate real-time glove control. Here, we take an intermediate approach and introduce a virtual joystick. Manipulation experiments with various effector loads and Cartesian trajectories indicate a 1.0 kg load limit in the fully extended mode and 1.5 kg limit in a midrange mode. This verifies vendor specifications for the arm and is acceptable for light-load daily mobility requirements.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

API	Application Program Interface
DOF	Degrees of Freedom
DH	Denavit-Hartenberg
GUI	Graphical User Interface
PC	Personal Computer
RML	Robotic Manipulation Laboratory
USB	Universal Serial Bus

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to thank my thesis advisor and second reader for their guidance and support. I would like to acknowledge my family for their support of this thesis work.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION AND OBJECTIVES

A robotic manipulation laboratory (RML) was implemented to help students understand tele-operation techniques with a sophisticated Kinova JACO robotic arm. The purpose of the research was to follow up on Jacinto's [1] work and expose students to the kinematics of robotic arm manipulation in a lab environment for various experiments. Jacinto demonstrated the viability of resistive glove control via simulation through the Robot Operating System (ROS) interface. He was not able to demonstrate real time glove control. Here we take an intermediate approach and introduce a virtual joystick.

The six degree of freedom JACO robotic arm is designed for both commercial and research applications. It enables users to interact with their environment in an intuitive and effective manner [2]. Our purpose was to characterize the arm performance under joystick control and then improve that with the development and demonstration of a virtual joystick. This required detailed knowledge of the 6-DOF Kinova JACO Arm link parameters and algorithms. Improvement in accuracy and smoothness of arm motion was the objective. These performance improvements are demonstrated by implementation of a block-stacking pyramid demonstration.

THIS PAGE INTENTIONALLY LEFT BLANK

II. THEORY

A time tested methodology to determine equations of motion for forward kinematic motion is to use Denavit-Hartenberg (DH) parameters to model robotic behavior. Here we review DH parameters as they apply JACO arm motion.

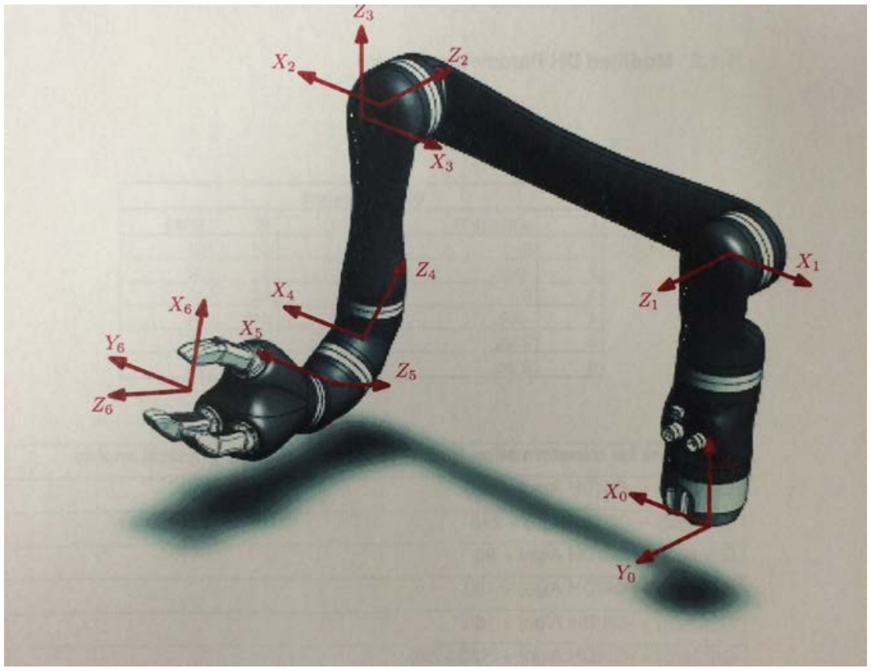
A. INTRODUCTION

By definition, DH parameters describe “the position and orientation of the links and joints that make up the robotic arm” [3].

Kinova provides DH parameters to the end user, as a means to move the end effector to a specific known position. Kinematics allows us to ignore forces, torques and inertias and simply concentrate on the position of the manipulator in space, regardless of the load of the manipulated object or the inertia of the arm while in motion [4].

The DH parameters are applied to the arm kinematic chain, where links are connected to joints and are driven by actuators. Each coupled joint-link establishes a degree of freedom (DOF). Consequently, with a 6-DOF arm, we find six links and six joints. The joints are numbered sequentially from the base to the end effector. The base (link 0) is excluded as a degree of freedom. Because the links keep a fixed relationship between the joints, frames of reference are used joint to joint to position the manipulator at a desired position by application of DH parameters and coordinate transformations (see Figure 1) [4].

Figure 1. Classic DH parameters frame position.



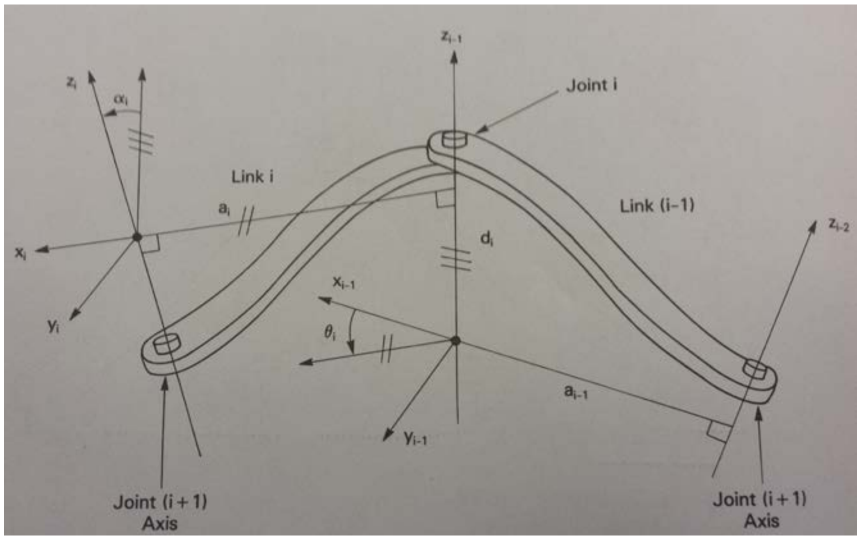
Source: [5] JACO *DH Parameters of JACO R&D*, V1.1.5, Kinova, Boisbriand, Canada, 2013.

B. DENAVIT-HARTENBERG PARAMETERS

The DH link coordinates and joint parameters are shown in Figure 2 and described as follows:

- For the link structure:
  - a_i (length of the link)
  - α_i (twist of the link).
- For relative position between links:
  - d_i (distance between the two x-z normals)
  - θ_i (angle between the two x normals) [4].

Figure 2. Link coordinate frame and joint parameters.



Source: [4] R. D. Klafter, T. A. Chmielewski and M. Negin, *Robotic Engineering: An Integrated Approach*, 1st ed. Englewood Cliffs, NJ: Prentice Hall, 1989.

The DH parameters are used to relate each link with neighboring links and to identify each link-joint pair. After the reference frames for each link are fixed, these parameters are used in a transformation matrix to connect successive frames [4].

To establish the link coordinate frames:

- The first rule is “the z_{i-1} axis lies along the axis of motion of the ith joint” [4].
- The second rule is “the x_i axis is normal to the z_{i-1} axis directed toward the z_i axis” [4].
- The third is that “the y_i axis is defined by the cross product of z_i and x_i, then the three axis form a right-handed system” [4].

Once the link parameters and references frames are determined, we can relate consecutives frames (i-1) to i, via translations and rotations, to correlate the two reference frames [4]. For example, see Equation 1 in [4].

$$A_{(i-1),i} = Rot_{z,\theta_i} Trans_{z,d_i} Trans_{x,a_i} Rot_{x,\alpha_i}$$

$$= \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & 0 \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_i & -\sin\alpha_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

$$= \begin{bmatrix} \cos\theta_i & -\sin\theta_i \cos\alpha_i & \sin\theta_i \sin\alpha_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\theta_i \cos\alpha_i & -\cos\theta_i \sin\alpha_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The matrix *A* is the transform from one joint to another via

- A rotation around z,
- A translation along d,
- Another translation along a
- And finally a rotation around x.

In the matrix, a_i and α_i are constant whereas d_i and θ_i are joint variables. The submatrix in blue depicts rotation, the submatrix in orange depicts translation, the perspective transform in red and the scaling factor is in green [4].

As discussed by Klafert section 8.3 of [4]:

The elements of the first column of the submatrix in blue are the components of the unit vector defining the x axis of frame 2, written in terms of the three unit vectors defining the axes of frame 1. The second column defines the y axis of frame 2 and the third column defines the direction of the z axis. The submatrix in orange is the location of the origin of frame 2 with respect to 1 and may be thought of as a vector drawn from the origin of frame 1 to the point defining the origin of 2. [4]

Finally, once the coordinate frames are established and the values for the DH matrices are known, we can determine the position of the JACO arm end effector with respect to the base via following transform [4].

$$T = A_{0n} = A_{01}A_{12}A_{23}...A_{(n-1)n} \tag{2}$$

Since the Kinova JACO arm is a 6-DOF manipulator, the expression is given by:

$$T = A_{01}A_{12}A_{23}A_{34}A_{45}A_{56} \tag{3}$$

To determine the final position of the end effector, the T matrix operates on the initial position vector as shown in equation 4 [1].

$$\text{Final Position} = T\begin{bmatrix} \mathrm{P}_x & \mathrm{P}_y & \mathrm{P}_z & 1 \end{bmatrix}^T \tag{4}$$

The following Tables 1 and 2 show the DH parameters of the Kinova JACO arm:

Table 1. Robot length values (meters).

D1	0.2755	Base to elbow
D2	0.4100	Arm length
D3	0.2073	Front arm length
D4	0.0743	First wrist length
D5	0.0743	Second wrist length
D6	0.1687	Wrist to center of the hand
e2	0.0098	Joint 3–4 lateral offset

Source: [5] JACO *DH Parameters of JACO R&D*, V1.1.5, Kinova, Boisbriand, Canada, 2013.

Table 2. DH parameters.

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	$\pi / 2$	0	D1	q1
2	π	D2	0	q2
3	$\pi / 2$	0	-e2	q3
4	2*aa	0	-d4b	q4
5	2*aa	0	-d5b	q5
6	π	0	-d6b	q6

Source: [5] JACO *DH Parameters of JACO R&D*, V1.1.5, Kinova, Boisbriand, Canada, 2013.

With the forward kinematic solution established, as represented in equation 4, we can locate the end effector in space given the values of "θ" and "α," since "d" and "a" are known.

C. INVERSE KINEMATICS

With knowledge of the Forward Kinematic Transform, we can place the robotic arm at a predetermined position. To do this, we look to the Inverse Kinematic Solution. This will tell us how to position each joint angle to get to the desired position and orientation. This calculation requires the multiplication of the forward Kinematic Transform with its inverse and then solving for the required joint angles. It is an iterative process and, for a 6 DOF system, will yield six equations that can be analytically solved for the required joint angles. For complex systems this is best handled with computer algorithms. For details on the process refer to Niku [6].

D. CARTESIAN TRAJECTORY PLANNING

Cartesian space trajectories are determined by repeatedly calculating the Inverse Kinematic Equations of the Robot and applying them iteratively to a chosen trajectory in Cartesian Coordinates. Niku indicates that this process is easily handled by computer code. His procedure is quoted as follows:

1. Increment the time by delta t.
2. Calculate the position and orientation of the hand based on selected function for the trajectory.
3. Calculate the joint values for the position and orientation through the inverse kinematic equations of the robot.
4. Send the joint information to the controller.
5. Go to the beginning of the loop. [6]

E. LAGRANGIAN DYNAMICS

When an understanding of forces, masses, loads and inertias are required to determine actuator performance requirements, it is quite natural to invoke Lagrangian Mechanics to understand the dynamics of the system.

$$L = T - U \tag{5}$$

In this equation L is the Lagrangian, T is the kinetic energy and U is the potential energy of the system. With the proper selection of generalized coordinates, the kinetic and potential energies can be determined. The equations of motion are derived via

$$\frac{\partial L}{\partial q_i} = \frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} \tag{6}$$

The q's are the generalized coordinates for our problem.

With the dynamic equations of motion, it is possible to predict the torques and forces on the arm for various arm trajectories under different loads. This information helps to determine actuator specifications for the robot arm.

THIS PAGE INTENTIONALLY LEFT BLANK

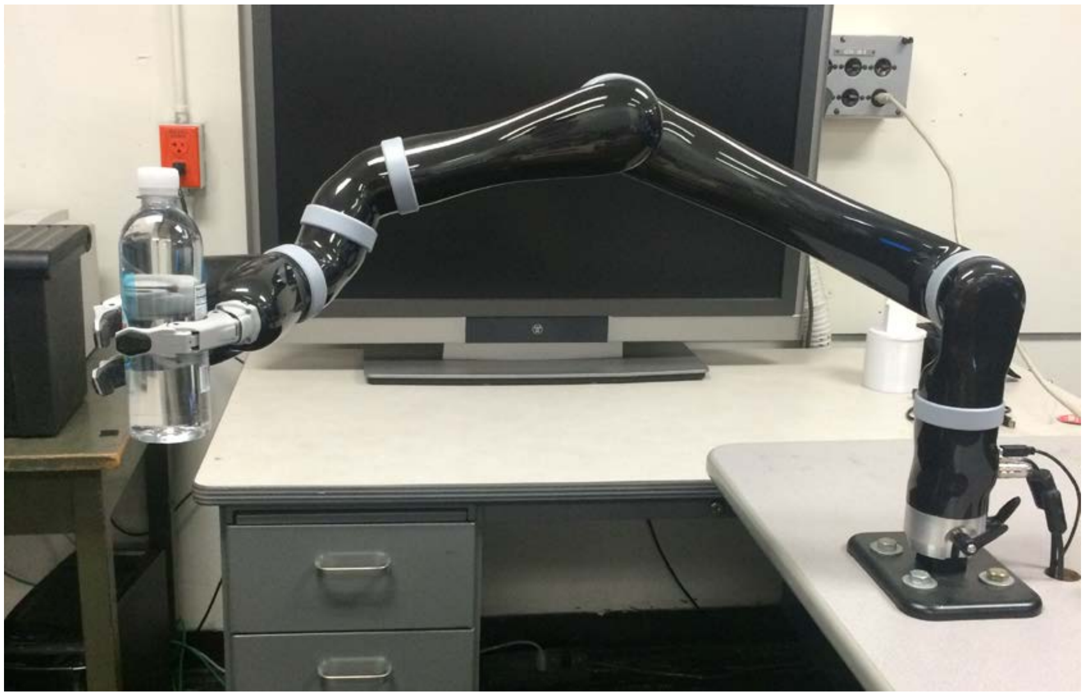
III. EXPERIMENTAL SETUP

In this chapter we characterize the JACO robotic arm and review the JACOSOFT software used to manipulate the robotic arm. User interaction with the lab is realized via a trajectory tab and a controller GUI with an API interface. Figures printed in 3D were used as manipulation objects throughout the experiment.

A. 6-DOF JACO ARM

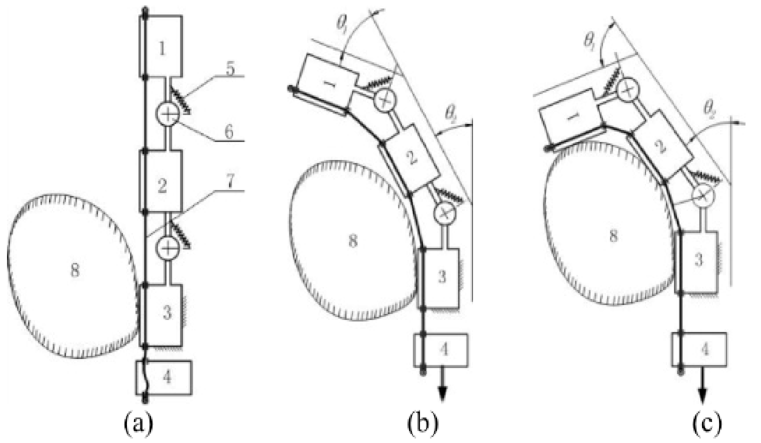
The JACO robotic arm is useful in various research applications. It is "a robotic manipulator that enables users to interact with their environment with safety, freedom, and effectiveness" [2]. The weight of the robotic arm is 5.6 kg with a maximum speed of 15 cm/s. The arm can be extended 90 cm in all directions while transporting weights up to 1.5 Kg [2] (see Figure 3).

Figure 3. JACO arm lifting a bottle.



The JACO Hand is an "Underactuated" gripper as opposed to a standard 2 finger industrial gripper. Underactuated grippers allow the finger joints to curl naturally like a human finger. This is accomplished by the arrangements of joint pins, "tendons" and return springs (see Figure 4).

Figure 4. The three stages for grasping with an Underactuated Gripper.



Stage one shown in part (a) is the initial stage. Stage two, part (b) is the pre-shaping stage and Stage Three, part (c) is the closing stage. Poorly tuned underactuated grippers can have difficulty grasping complex objects from. Source: [7] Long Wang, Joseph DelPetro et al. "Highly-underactuated robotic hand with force and joint angle sensors."

The JACO arm has environmental, electrical, mechanical and firmware specifications, as shown in Table 3.

Table 3. JACO arm specifications.

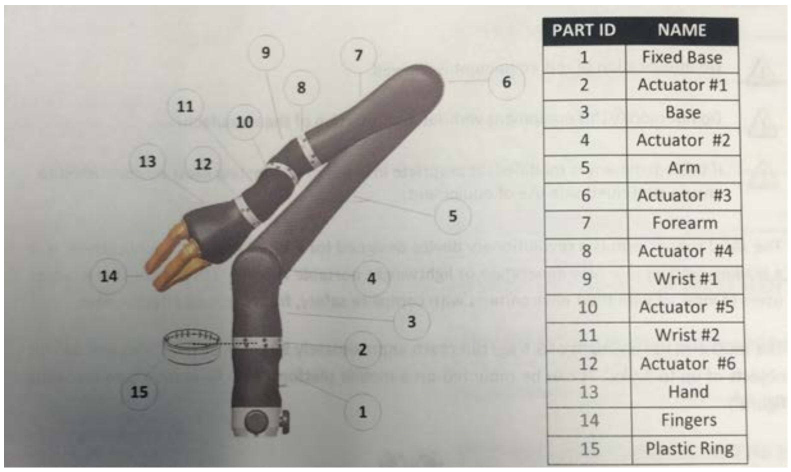
SPECIFICATIONS
ENVIRONMENT **GENERAL** Ambient temperature from 0 to 30 C degrees. Can be used under light rainfall for limited period. Can be used under normal atmospheric pressure conditions. **STORAGE** Ambient temperature from 0 to 50 C degrees. Relative Humidity: 55 % max.
ELECTRICAL **INPUT POWER** Voltage: 18V to 29V d.c. Current: 2A in normal use, 10A max. Powered by the external power supply. **OUTPUT POWER (controller port)** Voltage: 24V +/- 20% d.c. Current: 1.5A continuous, 3A max. **POWER SUPPLY** Input power: 100V to 240V a.c., 50Hz to 60 Hz, 2.0A Output power: 24 d.c, 5.0A
MECHANICAL **GENERAL** Total weight: 5.6 Kg +/- 5% Maximum Load: 1.5 Kg at mid-range (45 cm) Maximum Load: 1.0 Kg at end-range (90 cm) Reach: 90 cm Maximum linear arm speed: 15 cm/s **HAND AND FINGERS** 3 or 2 fingers simultaneous utilization. Independent control available for each finger.

Finger force limited to 7N (1.54 lb_f).
Flexible fingers for durability.
FIRMWARE
Each axis controlled independently.
Redundant security on each axis/fingers.
Redundant error check in joints and in control system.
Position and error calculation every 0.01 second.
Automatic recovery on system fault.

Source: [2] JACO *Research Edition User Guide*, V1.0.0. Kinova, Boisbriand, Canada, 2011–2012.

The six arm joints have limitless rotation. Each joint has an "actuator that integrates DC brushless motors, a harmonic drive, slip rings, a microcontroller, and sensors for torque, current, position, temperature, and acceleration" (see Figure 5) [8]. Six DC geared servomotors drive each joint. Joints 1 to 3 work with large motor modules while joints 4 to 6 are driven with small motor modules. An individual motor drive is mapped to each finger of the end-effector [9].

Figure 5. JACO arm actuators.



PART ID	NAME
1	Fixed Base
2	Actuator #1
3	Base
4	Actuator #2
5	Arm
6	Actuator #3
7	Forearm
8	Actuator #4
9	Wrist #1
10	Actuator #5
11	Wrist #2
12	Actuator #6
13	Hand
14	Fingers
15	Plastic Ring

Source: [10] JACO *JACOSOFT User Guide*, V1.4.2, Kinova, Boisbriand, Canada, 2011–2014.

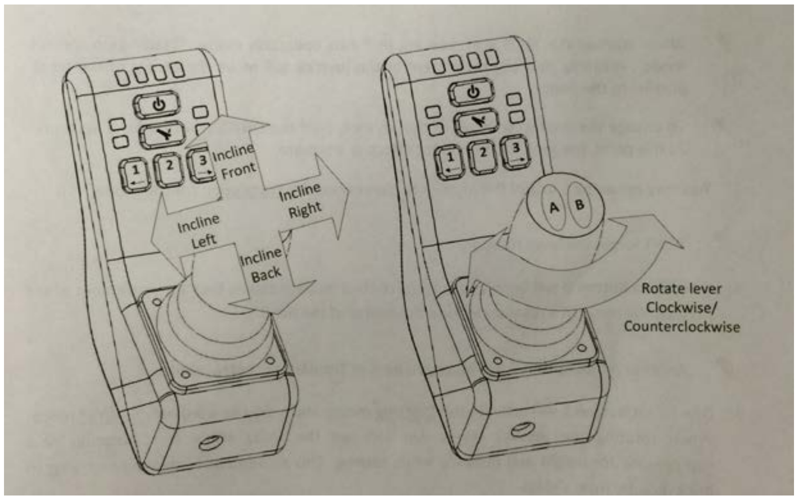
The robotic arm comes with a 3-axes joystick. Modes of operation include:

1) Incline left/right,

2) Incline front/back,

3) Rotation of the lever clockwise/counter-clockwise (see Figure 6) [2].

Figure 6. Kinova's joystick possible commands.



Source: [2] JACO *Research Edition User Guide*, V1.0.0. Kinova, Boisbriand, Canada, 2011–2012.

B. JACOSOFT

JACOSOFT software allows the user to monitor robot arm device information and to set various arm use configurations. This section describes the Configurations and Trajectory tabs.

1. Configurations Tab

The Configurations tab includes two sub tabs: Positioning and Control.

a. Positioning

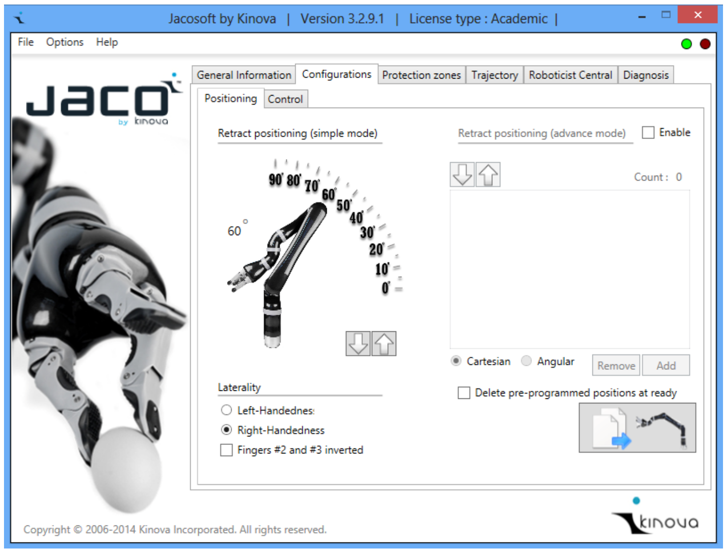
The Positioning tab allows the user to set the following parameters:

- Laterality, left or right handedness [10].

- Retracted angle, the retracted angle of the RETRACT position [10].
- Retracted trajectory, the HOME position, the RETRACT position and the trajectory between those positions [10].

Figure 7 shows the Configurations/Positioning tab [10].

Figure 7. Configurations/Positioning tab.



Retract positioning (simple mode)

60°

Laterality

( ) Left-Handednes

(●) Right-Handedness

☐ Fingers #2 and #3 inverted

Retract positioning (advance mode) ☐ Enable

Count : 0

(●) Cartesian ( ) Angular

☐ Delete pre-programmed positions at ready

Source: [10] JACO *JACOSOFT User Guide*, V1.4.2, Kinova, Boisbriand, Canada, 2011–2014.

b. Control

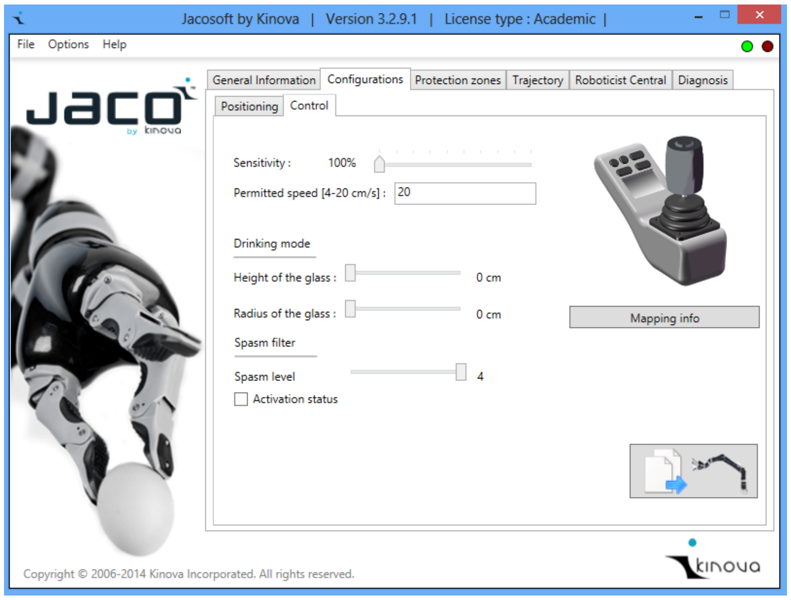
The Control tab gives the user the ability to set the following parameters:

- The joystick sensitivity between values of 0 to 100% [10].
- The linear arm speed between 4 and 20 cm/s [10].
- Drinking mode. The drinking mode parameters include the Height of the glass/bottle (up to 10 cm) and the Radius of the glass/bottle (up to 6 cm) [10].
- Spasm filter. This filter is used to reduce or eliminate involuntary commands sent to the arm. Filters can be set to values from 0 to 4. (0-Not activated, 1-Very low filter, 2-Low filter, 3-Medium filter, 4-High filter). [10].

- Controller setting configuration Map. See JACOSOFT User Guide [10] for detailed information.

Figure 8 shows the Configurations/Control tab [10].

Figure 8. Configurations/Control tab.



Jacosoft by Kinova | Version 3.2.9.1 | License type : Academic |

Sensitivity : 100%

Permitted speed [4-20 cm/s] : 20

Drinking mode

Height of the glass : 0 cm

Radius of the glass : 0 cm

Spasm filter

Spasm level 4

☐ Activation status

Copyright © 2006-2014 Kinova Incorporated. All rights reserved.

Source: [10] JACO *JACOSOFT User Guide*, V1.4.2, Kinova, Boisbriand, Canada, 2011–2014.

3. Trajectory

The trajectory tab allows the user to configure, save and load a trajectory. The trajectory is invoked in either Cartesian or Angular control [10]. Saved trajectories move the arm, point to point, for the planned path of the JACO Arm.

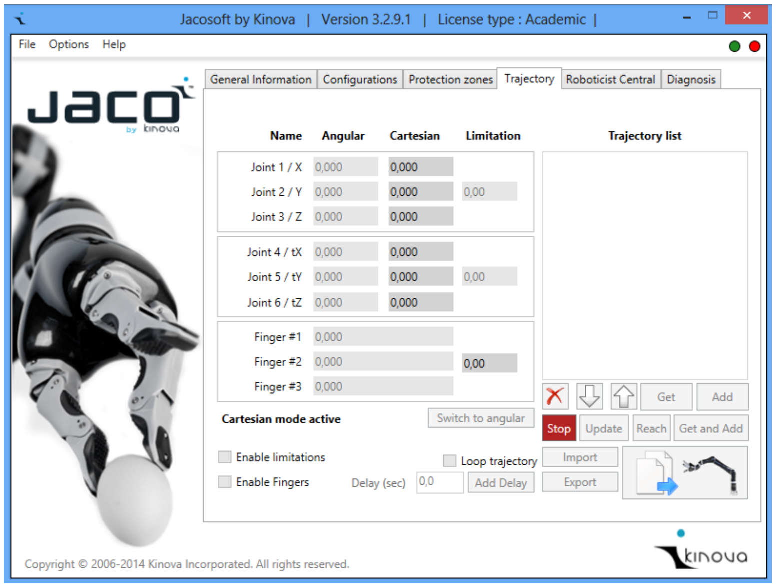
In the Trajectory tab, the user sets the following parameters:

- Trajectory mode: Cartesian or Angular [10].
- Speed limitations [10].
- Enable Fingers: Set for finger manipulation in a trajectory [10].
- Trajectory delay. [10].

- Loop trajectory mode: on/off. [10].
- Get, add, stop, update, reach and delete trajectories from the trajectory list [10].
- Import and export trajectories [10].

Figure 9 shows the Trajectory tab [10].

Figure 9. Trajectory tab from.



Name	Angular	Cartesian	Limitation
Joint 1 / X	0,000	0,000	
Joint 2 / Y	0,000	0,000	0,00
Joint 3 / Z	0,000	0,000	
Joint 4 / tX	0,000	0,000	
Joint 5 / tY	0,000	0,000	0,00
Joint 6 / tZ	0,000	0,000	
Finger #1	0,000		
Finger #2	0,000		0,00
Finger #3	0,000		

Cartesian mode active

☐ Enable limitations

☐ Enable Fingers

☐ Loop trajectory

Delay (sec) 0,0

Copyright © 2006-2014 Kinova Incorporated. All rights reserved.

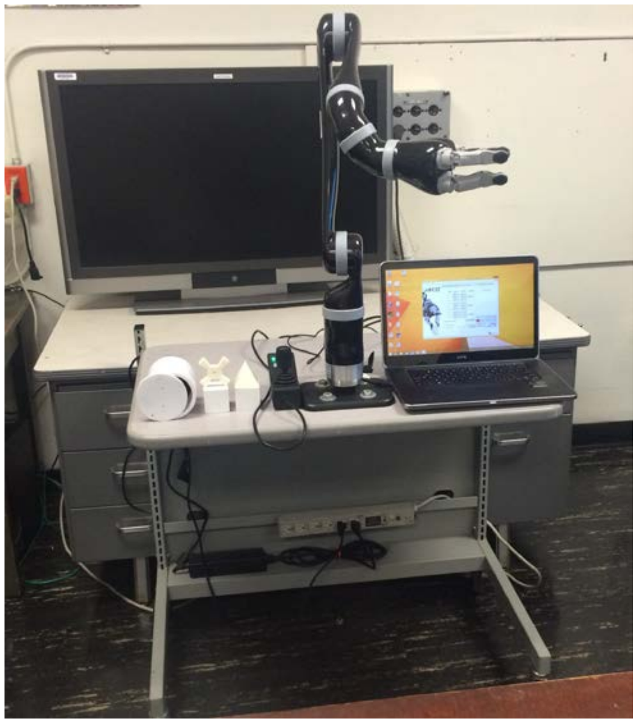
Source: [10] JACO *JACOSOFT User Guide*, V1.4.2, Kinova, Boisbriand, Canada, 2011–2014.

C. JOYSTICK TRAJECTORY DEMO

The joystick demonstration included the steps to plan, save and load a Cartesian trajectory. The objective was to move objects from specified positions to alternate positions within the grasping length of the arm.

For this demonstration we used the JACO Kinova arm, the Kinova joystick, a laptop, the JACOSOFT software and 3D printed figures such as: a hollow cylinder with a cap, two cubes, a pyramid and a mini-wheg. The JACO arm was fastened to a portable table. An electric bus attached to the table was used to power the arm, the laptop and additional equipment as necessary (see Figure 10).

Figure 10. JACO arm with equipment implemented for Joystick trajectory demo.



The Kinova JACO arm has various external connectors including: a Power Connector, a Control Port and an USB Port. The Kinova joystick was connected to the JACO arm by the Control Port with a serial cable. The arm was connected to the laptop via a USB cable as shown in Figure 9.

The 3D printed objects were placed on the table in specific positions (see Figure 11). To build the trajectory, the joystick was used to move the arm. This was saved as a file via the Trajectory tab. For the demonstration, this trajectory was replayed as a pyramid block stacking routine by grasping and moving objects in a preplanned fashion (see Figure 12).

Figure 11. Initial position of objects.

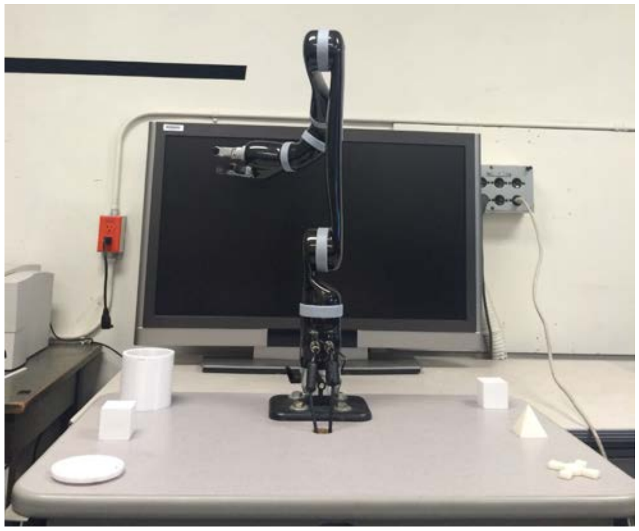
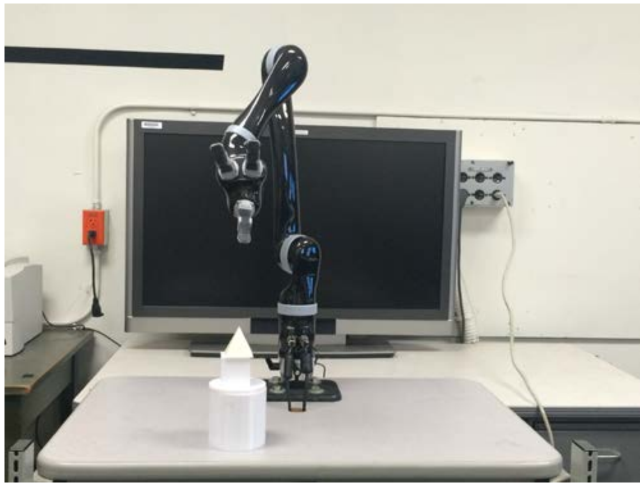


Figure 12. Final position of the objects.



Once saved, the trajectory can be replayed at will without the use of the joystick. This is very similar to a robotic arm used in an assembly line. For details on how to load and save a given trajectory, see Appendix A.

D. VIRTUAL JOYSTICK DEMO

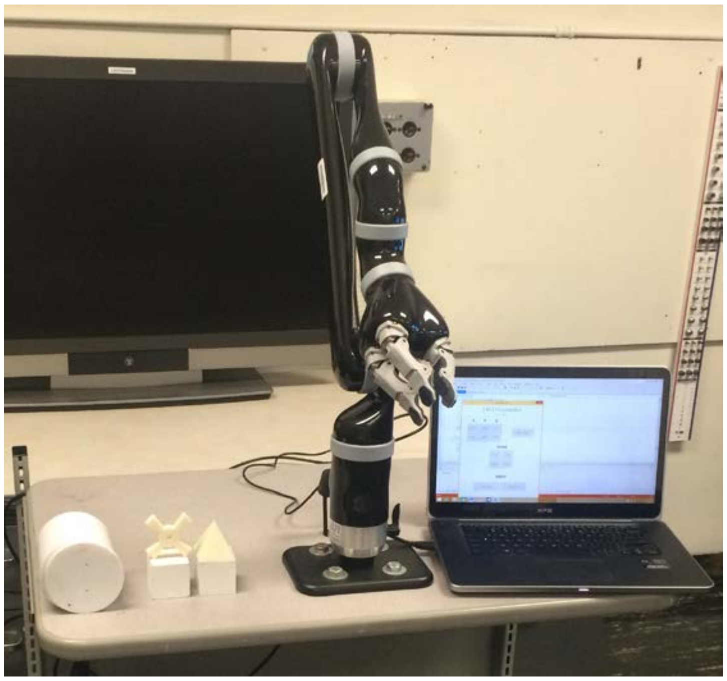
To eliminate the need for a mechanical Joystick, a Virtual Joystick was developed using the JACO Controller GUI with an API interface written in C# [11]. The GUI was modified with added buttons so the user can grasp and move objects from specific positions to required positions by a Virtual Joystick.

The lab setup included the JACO Kinova arm, a laptop, the JACOSOFT software, the JACO controller GUI and 3D printed figures. The figures were made up of

- a hollow cylinder with a cap,
- two cubes,
- a pyramid, and
- a wheg.

The equipment was implemented as shown in Figure 13.

Figure 13. JACO arm with equipment implemented for Virtual Joystick demo.

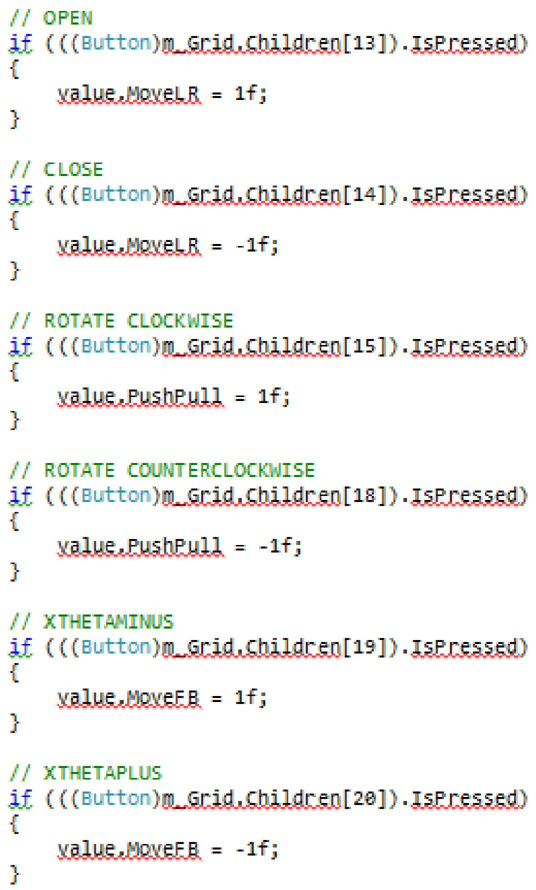


The JACO arm interfaces with the laptop via JACOSOFT software. Here the user is able to set the conditions for arm manipulation with the JACO Controller. The

“Mapping Info Window” in the API makes it possible to interface with the arm and the laptop using Visual Studio C#.

The C# code generates a Virtual Joystick (see Figure 14), a GUI that allows the user to manipulate the arm. To conduct the experiment, the 3D printed objects were placed in specific positions (see Figure 10) and were grasped by the arm and moved to a block stacking pyramid (see Figure 14).

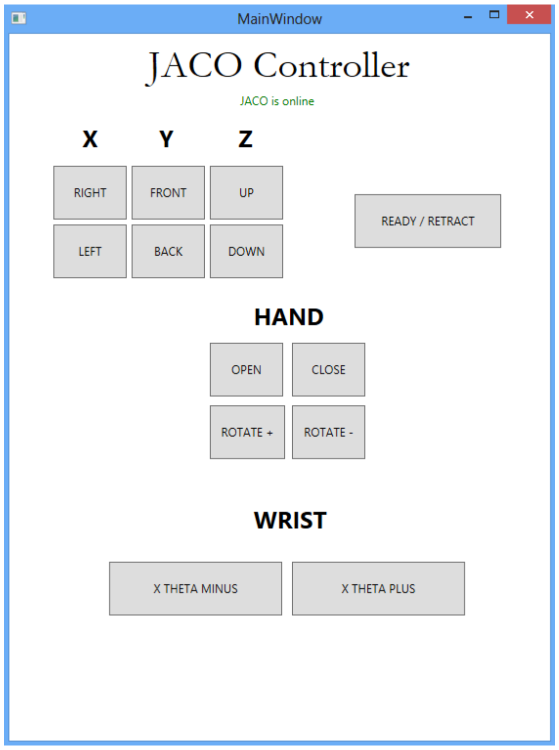
Figure 14. C# code for Virtual Joysticks buttons.



```
// OPEN
if (((Button)m_Grid.Children[13]).IsPressed)
{
    value.MoveLR = 1f;
}

// CLOSE
if (((Button)m_Grid.Children[14]).IsPressed)
{
    value.MoveLR = -1f;
}

// ROTATE CLOCKWISE
if (((Button)m_Grid.Children[15]).IsPressed)
{
    value.PushPull = 1f;
}

// ROTATE COUNTERCLOCKWISE
if (((Button)m_Grid.Children[18]).IsPressed)
{
    value.PushPull = -1f;
}

// XTHETAMINUS
if (((Button)m_Grid.Children[19]).IsPressed)
{
    value.MoveFB = 1f;
}

// XTHETAPLUS
if (((Button)m_Grid.Children[20]).IsPressed)
{
    value.MoveFB = -1f;
}
```

The Virtual Joystick is a graphical user interface used to manipulate the JACO arm in all directions (see Figure 15). The GUI is divided into three zones: Arm, Hand and Wrist. In the arm zone the buttons allow the user to move the arm from retract to home position, up/down, left/right and front and back. The buttons in the hand zone are used to open/close the hand and rotate it in the clockwise or counterclockwise direction. The wrist zone allows the user to point the hand up or down. The detailed process for JACO Arm manipulation is located in Appendix B.

Figure 15. Virtual Joystick (GUI).



THIS PAGE INTENTIONALLY LEFT BLANK

IV. EXPERIMENTAL RESULTS

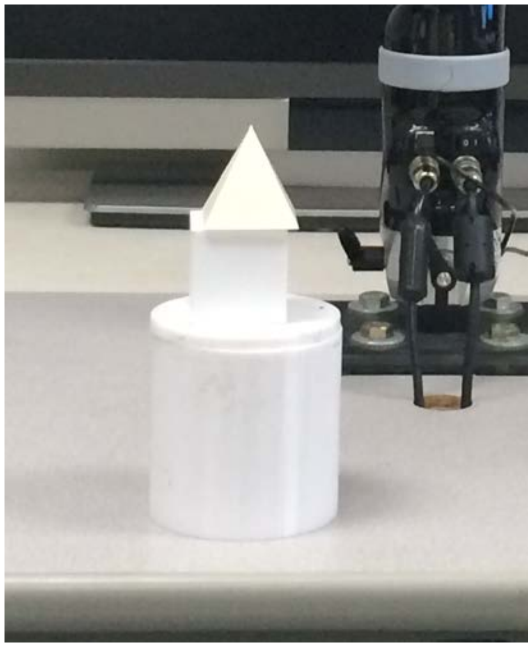
This section describes the results from the Joystick Trajectory Lab, the Virtual Joystick Lab and the performance of the JACO Arm as related to the force and current requirements to lift objects of different weight.

A. TRAJECTORY AND VIRTUAL JOYSTICK DEMO

Arm performance was demonstrated under Joystick and Virtual Joystick control for a predetermined Cartesian trajectory. The objective was to pick up and manipulate objects with varying degrees of complexity to test load and dexterity performance.

- Qualitatively the trajectory path performance was smooth for both experiments.
- The robotic arm was accurate over several block stacking runs. Small variations in placement position were observed as indicated by Figure 16.

Figure 16. Virtual Joystick (GUI).



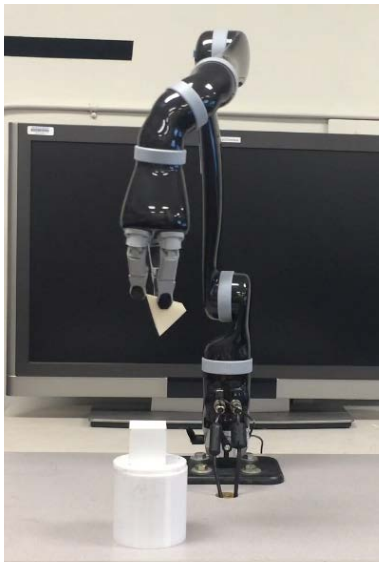
- A test moving a cube from a specific position to an alternate position was realized 10 times, showing the following data (see Table 4) from the distance between positions. The standard deviation for these attempts was 0.036 cm. It demonstrates the accuracy but with small variations in placement positions on the block stacking pyramid.

Table 4. Attempts to observe variations moving a cube from a specific position to an alternate position.

ATTEMPTS	DISTANCE (cm)
A1	32.512
A2	32.512
A3	32.461
A4	32.537
A5	32.461
A6	32.563
A7	32.461
A8	32.537
A9	32.487
A10	32.461

- Joystick operation was intuitive for what would be considered normal human actions. Primarily the movement of objects.
- Pyramid shapes, as shown in Figure 17, were difficult to manage with the current version of the three-finger end effector. This is attributed to a poorly tuned “underactuated” grasp technique.

Figure 17. JACO arm lifting a pyramid.

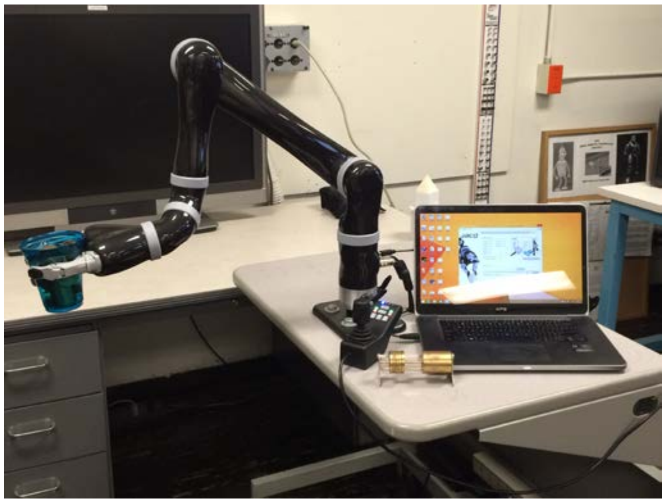


B. JACO ARM LOAD CHARACTERIZATION

To verify the capability of the arm to lift various loads, tests were conducted with the following weights: 0Kg, 0.5Kg, 1Kg, 1.5Kg and 2Kg (see Figures 18 and 19). Using the Virtual Joystick and the Health Center window, data was collected on torques and currents associated with the joint actuators for joints 2 and 3. These data were collected for arm lifts of 1/2 meter and 1 meter, respectively. Results are shown in Tables 4–7 and Figures 20–23.

Figure 18. Weights to be lifted by the JACO arm.

Figure 19. JACO arm to lift weights implementation.



Tables 2 through 5 display the data for JACO arm joints 2 and 3 for various weights for lift heights of 1/2m and 1m.

Table 5. Joint 2 torque and current for lifts of 0.5m.

WEIGHTS (kg)	TORQUE (Nm)	CURRENT (A)
0	12.6049	1.1459
0.5	16.4844	1.4986
1	19.5445	1.7768
1.5	22.4098	2.0373
2	25.8586	2.3508

Table 6. Joint 3 torque and current for lifts of 0.5m.

WEIGHTS (kg)	**TORQUE (Nm)**	**CURRENT (A)**
0	0	0
0.5	0	0
1	0.7052	0.0683
1.5	1.2046	0.1095
2	6.1390	0.5581

Table 7. Joint 2 torque and current for lifts of 1.0m.

WEIGHTS (kg)	**TORQUE (Nm)**	**CURRENT (A)**
0	11.4093	1.0372
0.5	13.5412	1.2310
1	17.7383	1.6126
1.5	22.4553	2.0414
2	30.2264	2.7479

Table 8. Joint 3 torque and current for lifts of 1.0m.

WEIGHTS (kg)	**TORQUE (Nm)**	**CURRENT (A)**
0	4.0103	0.3646
0.5	4.3014	0.3853
1	4.5775	0.4161
1.5	14.2720	1.2975
2	17.6919	1.6084

The experiment was designed to isolate joints 2 and 3 for the lift. The data is plotted in Figures 16 through 19. We notice that for the both the intermediate (0.5m) lift and the high (1.0m) lift, joint 2 is fully engaged for the duration of the lift for both heights with torque values ranging from 12–30 Newton meters. However, joint 3 did not engage until the load weight increased to over 1 kg. Current and torque loads were much lower for joint 3 in all cases. This is attributed to the geometry of the joint positions for the experiment. The conclusion is that torque values are definitely a function of joint starting position but load weights have a much more significant effect. Furthermore, current limits were exceeded on joint 2 for both the intermediate and high lift when the load weight exceeded 1.5 kg. The arm handled the load successfully but the long term risk is damage to the actuator.

Figure 20. Weight vs. torque for arm joints 2 and 3 for lifts of 0.5m.

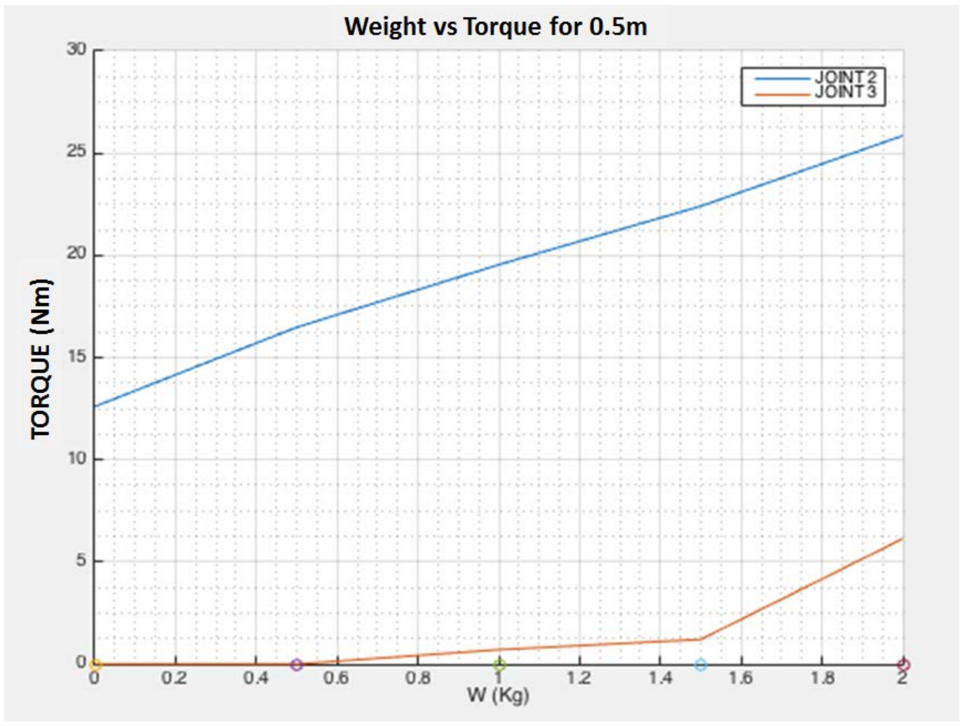


Figure 21. Weight vs. current for arm joints 2 and 3 for lifts of at 0.5m.

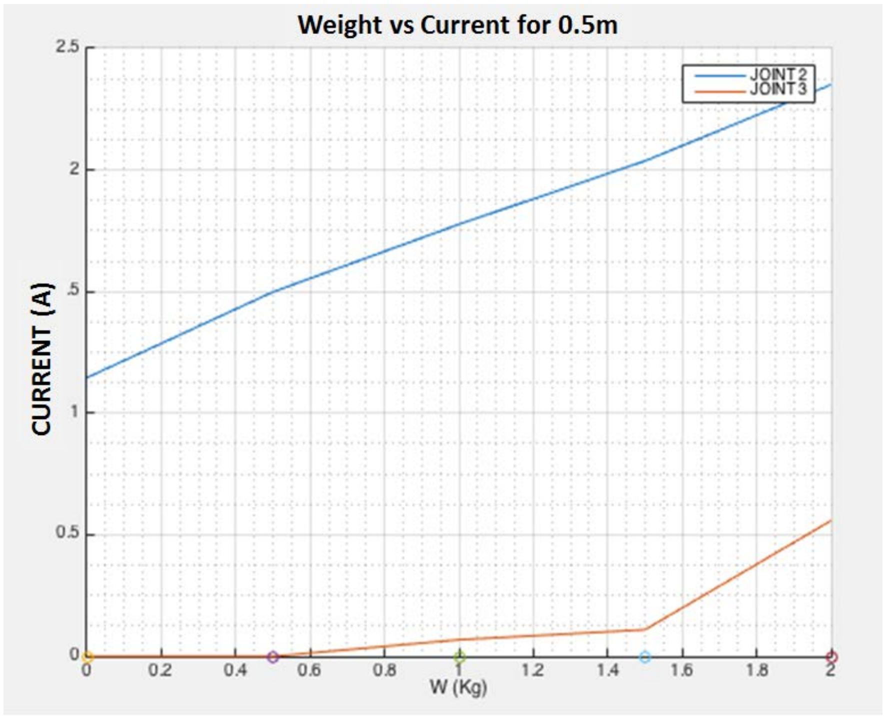


Figure 22. Weight vs. torque for JACO arm joints 2 and 3 for lifts at 1m.

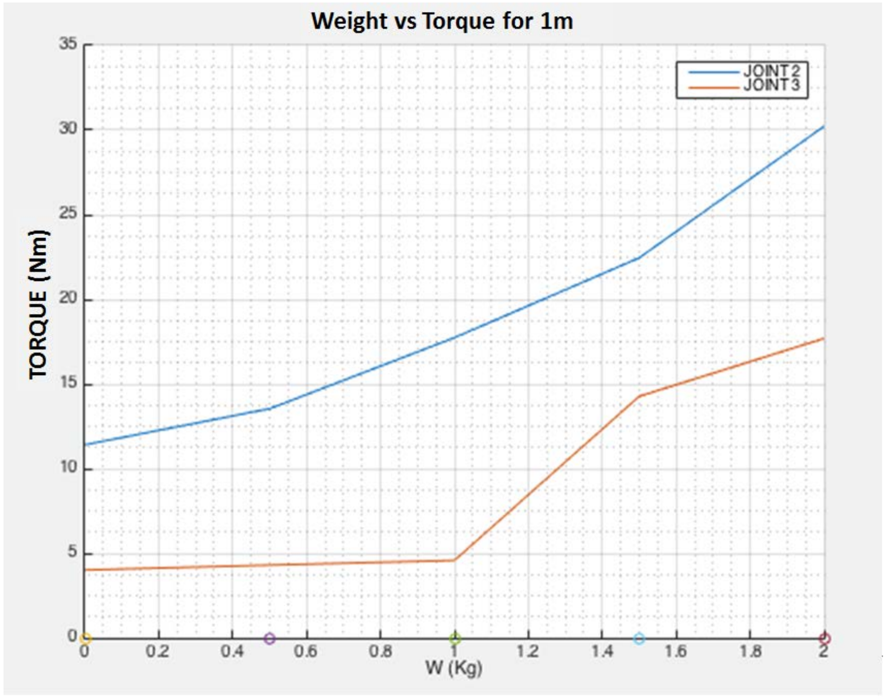
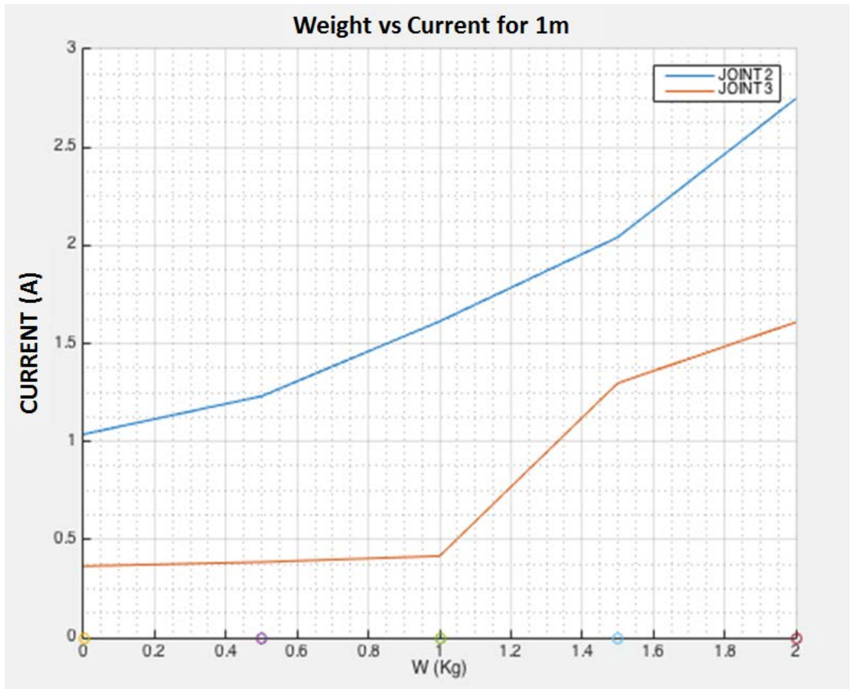


Figure 23. Weight vs. current for JACO arm joints 2 and 3 for lifts at 1m.



V. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

Based on observations with regard to JACO Arm performance under Joystick and Virtual Joystick and upon experimental load lift data the following summarizes our conclusions:

- Arm interaction with the Virtual Joystick, as opposed to the manual Joystick, is a preferred method primarily because it is simpler to handle repetitive actions and it is easier to load preplanned trajectories.
- The Visual Studio API interface is a viable intermediate step for future external interface capabilities with the JACO arm.
- Maximum lift capabilities, as listed in the technical specifications for the arm, were validated by experiment. For most lifting requirements, loads should never exceed 1.5 kg.

B. RECOMMENDATIONS

The JACO arm-API-Visual Studio-Virtual Joystick interface is the gateway to a new device interaction. This is because the API is grounded in Visual Studio. Many sophisticated motion capture devices are supported by Visual Studio. This includes Kinect for Windows, the Leap Motion Controller and the Myo Gesture Control Armband for devices supported by Arduino. The follow-on work should include an exploration of the integration of these devices with the JACO Arm.

Finally, the NPS AXV lab will explore future options to deploy this arm on an autonomous platform.

THIS PAGE INTENTIONALLY LEFT BLANK

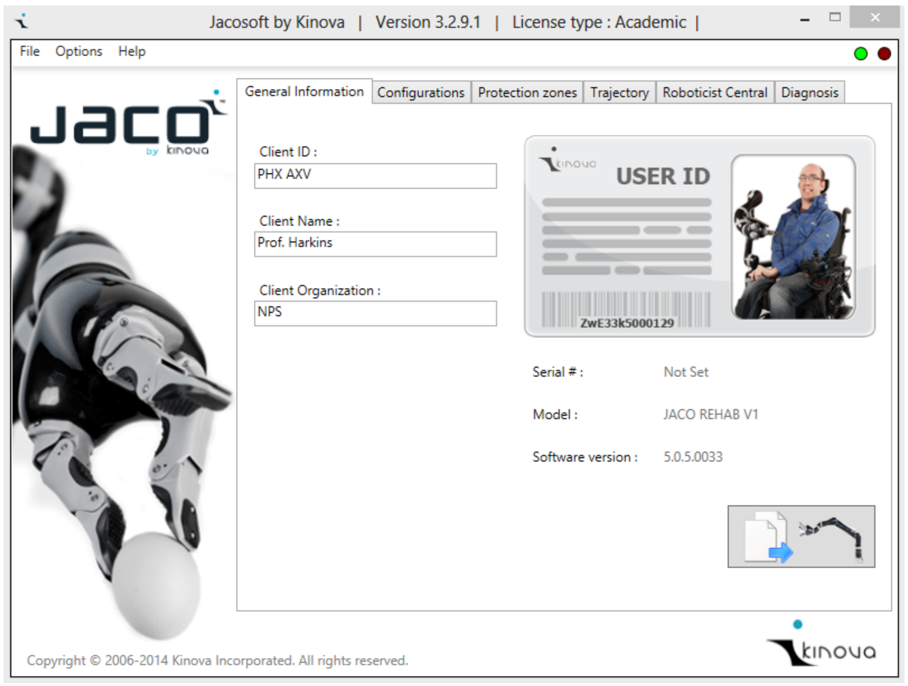
APPENDIX A

This section will describe the steps to save and load a specific trajectory in order to grasp and move objects from a specific position to another required position. This will help to facilitate student's experiments in a robotic manipulation laboratory (RML).

Once the arm is powered and connected via USB with a computer, based on [10], the steps are made, for saving and loading a specific trajectory for grasping and moving the 3D printed objects, are:

Open the JACOSOFT software. Once it loads the Client ID, Client Name and Client Organization in the General Information tab, it will be ready to use (see Figure 24).

Figure 24. General Information tab. JACOSOFT ready to use.



Client ID : PHX AXV

Client Name : Prof. Harkins

Client Organization : NPS

Serial # : Not Set

Model : JACO REHAB V1

Software version : 5.0.5.0033

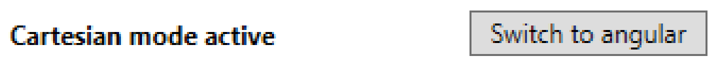
Copyright © 2006-2014 Kinova Incorporated. All rights reserved.

Source: [10] JACO *JACOSOFT User Guide*, V1.4.2, Kinova, Boisbriand, Canada, 2011–2014.

Figure 25. Cartesian mode active. Switch to angular button in Trajectory tab.

Click on Trajectory tab.

Choose a Cartesian or Angular mode. The software is set by default in Cartesian mode. User can switch the mode by clicking the button **Switch to angular** (see Figure 25). For this lab setup the JACO arm will work in Cartesian mode. **Cartesian mode active** will be showed on the window.



Click on the box **Enable Fingers**. It will allow the software get the position of the fingers in order to open and close it to grasp and move objects (see Figure 26).

Figure 26. Enable Fingers box.

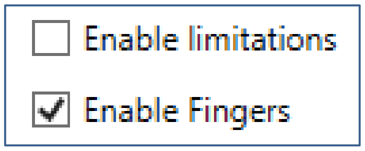
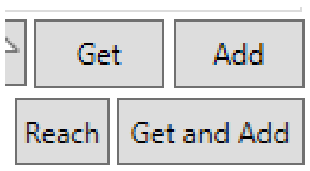


Figure 27. GET and ADD button.

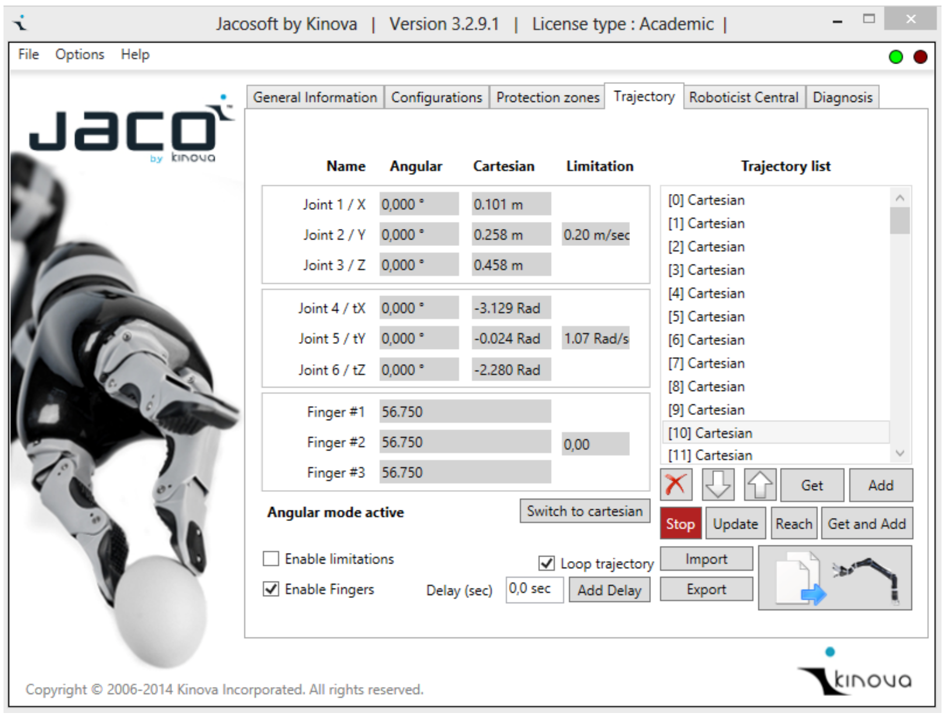
Move the arm to the **HOME position** using the joystick, in order to start to save the trajectory from an established position.

Click the **GET and ADD** button in order to save the trajectory (see Figure 27). The HOME position will be the first saved trajectory. The first trajectory will appear in the **Trajectory list** box as **[0] Cartesian** (see Figure 28).



Using the joystick, go to the next position of the arm and fingers to grasp the first object and then click on the button **GET and ADD** to save the trajectory. Move the object to the required position using the joystick and save the new trajectory. The user will repeat the process until all the objects are moved to the required positions. The list of trajectories will appear in the **Trajectory list** window (see Figure 28).

Figure 28. Trajectory list box in Trajectory tab.



Jacosoft by Kinova | Version 3.2.9.1 | License type : Academic |

File Options Help

General Information | Configurations | Protection zones | Trajectory | Roboticist Central | Diagnosis

Name	Angular	Cartesian	Limitation
Joint 1 / X	0,000 °	0.101 m	
Joint 2 / Y	0,000 °	0.258 m	0.20 m/sec
Joint 3 / Z	0,000 °	0.458 m	
Joint 4 / tX	0,000 °	-3.129 Rad	
Joint 5 / tY	0,000 °	-0.024 Rad	1.07 Rad/s
Joint 6 / tZ	0,000 °	-2.280 Rad	
Finger #1	56.750		
Finger #2	56.750		0,00
Finger #3	56.750		

Trajectory list

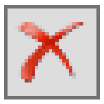
[0] Cartesian
[1] Cartesian
[2] Cartesian
[3] Cartesian
[4] Cartesian
[5] Cartesian
[6] Cartesian
[7] Cartesian
[8] Cartesian
[9] Cartesian
[10] Cartesian
[11] Cartesian

Angular mode active

☐ Enable limitations
☑ Enable Fingers
☑ Loop trajectory
Delay (sec) 0,0 sec

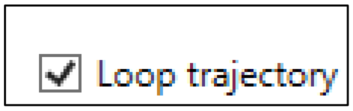
To delete a trajectory, the user should select one such as [3] Cartesian and click on the **X** button (see Figure 29).

Figure 29. X button.



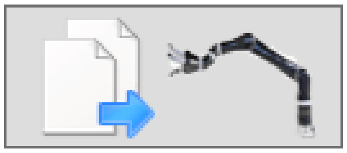
Click on the box **Loop trajectory** for repeat the whole trajectory indefinite times (see Figure 30).

Figure 30. Loop trajectory box.



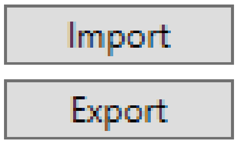
Click on **SEND** button to start the trajectory (see Figure 31).

Figure 31. SEND button.



To save the trajectory, click on **Export** button. Click on **Import** button to load a saved trajectory (see Figure 32).

Figure 32. Import and Export buttons.



This lab setup to grasp and move objects based on joystick and trajectory tab used 97 trajectories to achieve the final specific position for all the objects. The following figures show the initial position of the objects (see Figure 33), some intermediate positions (see Figure 34 to 36) and the final position (see Figure 37).

Figure 33. Initial position of objects.

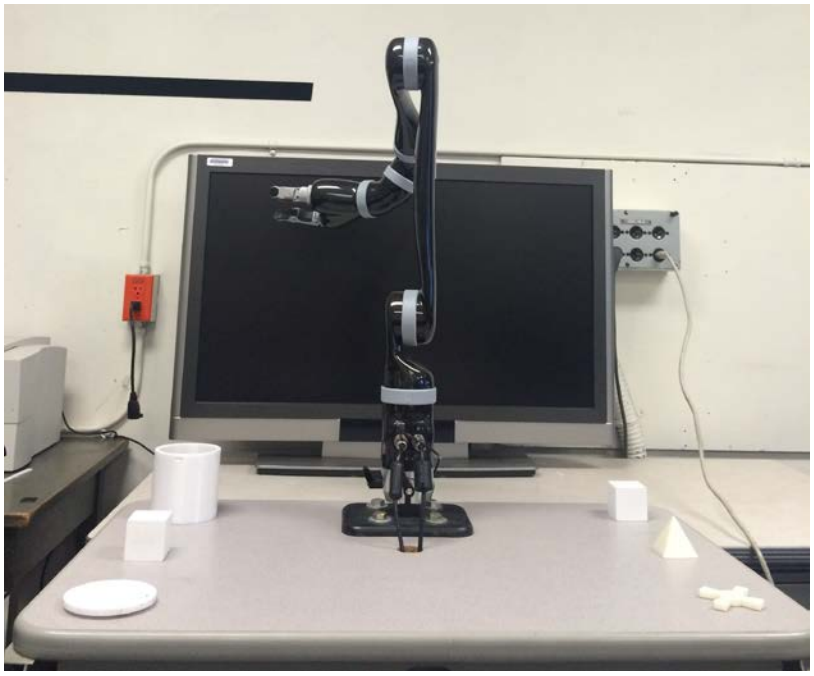


Figure 34. Manipulating cube inside cylinder.

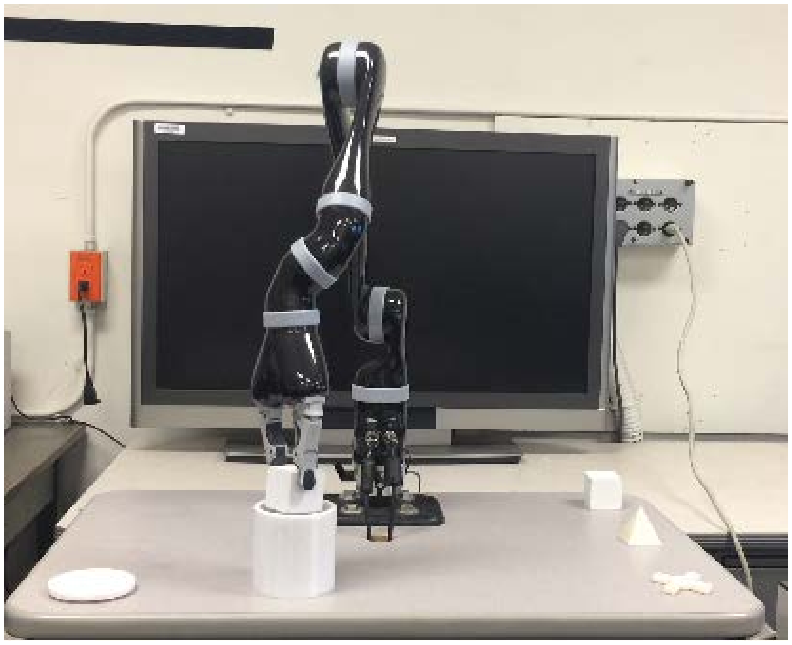


Figure 35. Covering the cylinder.

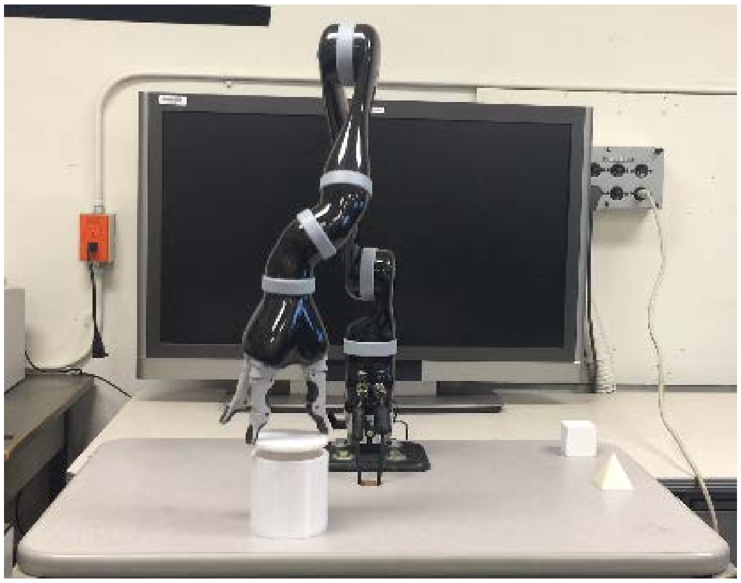


Figure 36. Manipulating pyramid on the cube.

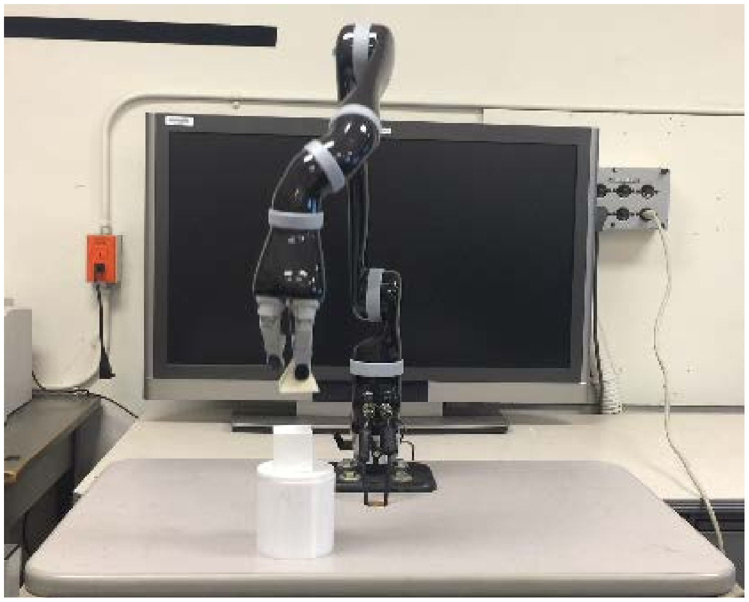
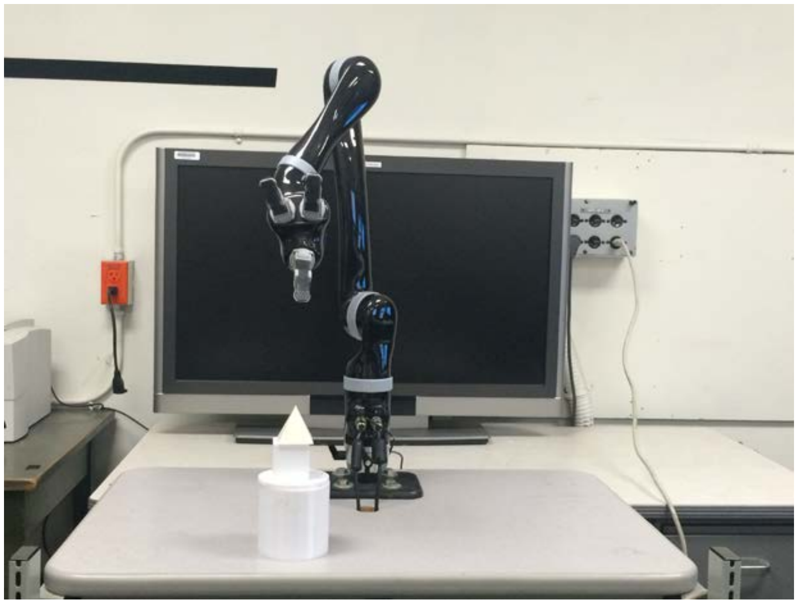


Figure 37. Final position of the objects.



THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B

The steps to follow in order to manipulate the JACO arm with the GUI are:

Once the arm is powered and connected via USB with a computer, based on [10], the steps to follow in order to manipulate the JACO arm with the GUI for grasp and move the printed objects, are:

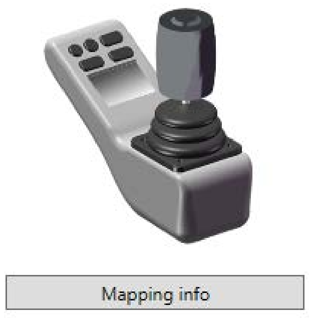
Open the JACOSOFT software.

Click on **Configurations** tab.

Click on **Control** sub tab.

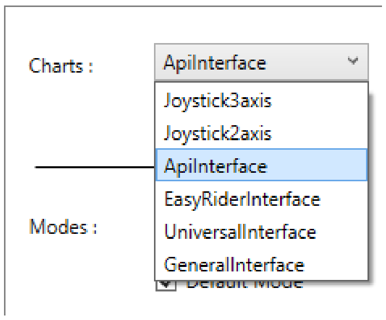
Click on button **Mapping Info** (see Figure 38).

Figure 38. Mapping info button.



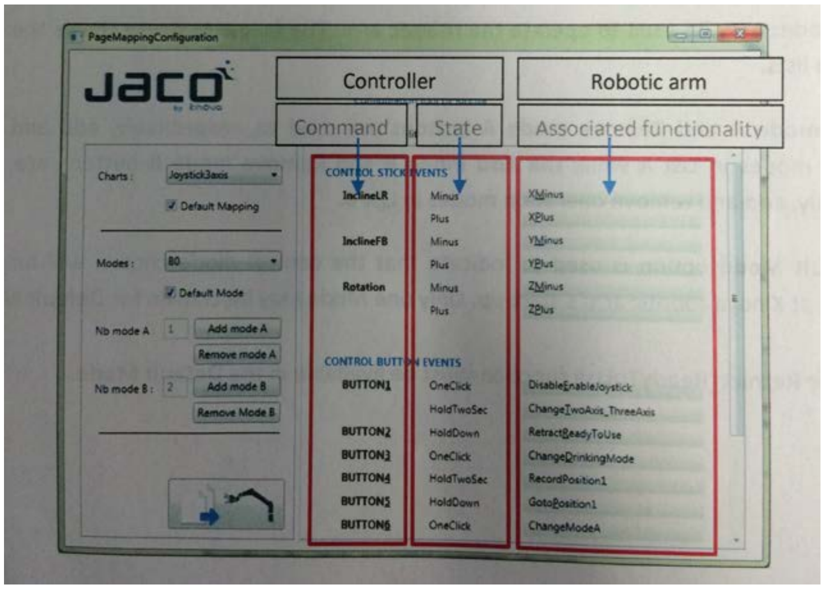
On Mapping window, in Charts select **ApiInterface** (see Figure 39).

Figure 39. ApiInterface option.



Set the controller’s functionalities (see Figure 40).

Figure 40. Command/State. Associated functionality.



Charts : Joystick3axis
☑ Default Mapping
Modes : 80
☑ Default Mode
Nb mode A : 1 Add mode A
Remove mode A
Nb mode B : 2 Add mode B
Remove Mode B

CONTROL STICK EVENTS

InclineLR	Minus	XMinus
	Plus	XPlus
InclineFB	Minus	YMinus
	Plus	YPlus
Rotation	Minus	ZMinus
	Plus	ZPlus

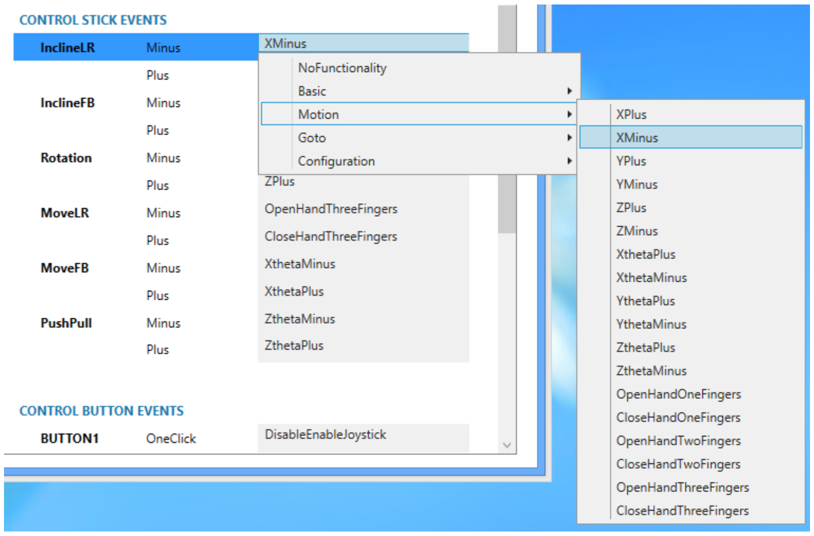
CONTROL BUTTON EVENTS

BUTTON1	OneClick	DisableEnableJoystick
	HoldTwoSec	ChangeTwoAxis_ThreeAxis
BUTTON2	HoldDown	RetractReadyToUse
BUTTON3	OneClick	ChangeDrinkingMode
BUTTON4	HoldTwoSec	RecordPosition1
BUTTON5	HoldDown	GotoPosition1
BUTTON6	OneClick	ChangeModeA

Source: [10] JACO *JACOSOFT User Guide*, V1.4.2, Kinova, Boisbriand, Canada, 2011–2014.

Select an Associated functionality/Motion for each Control Stick Event (Command/State) (see Figures 40 and 41).

Figure 41. Associated functionality/motion.



CONTROL STICK EVENTS

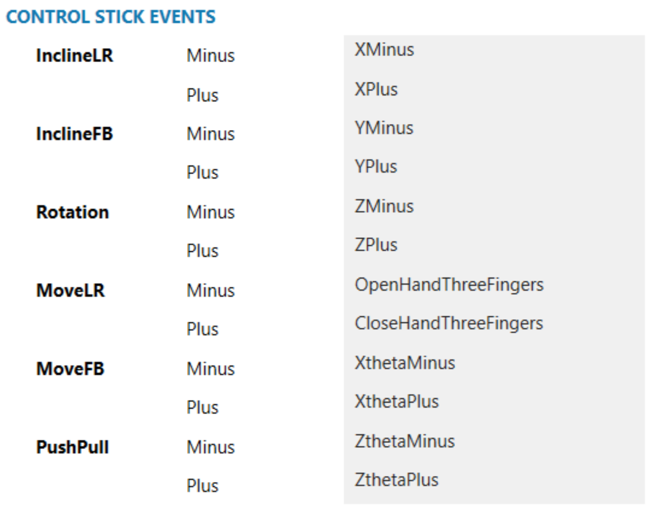
InclineLR	Minus	XMinus
	Plus	
InclineFB	Minus	
	Plus	
Rotation	Minus	
	Plus	ZPlus
MoveLR	Minus	OpenHandThreeFingers
	Plus	CloseHandThreeFingers
MoveFB	Minus	XthetaMinus
	Plus	XthetaPlus
PushPull	Minus	ZthetaMinus
	Plus	ZthetaPlus

CONTROL BUTTON EVENTS

BUTTON1	OneClick	DisableEnableJoystick

Figure 42 shows the Associated functionalities/Motion selected for this lab setup.

Figure 42. Control stick events.

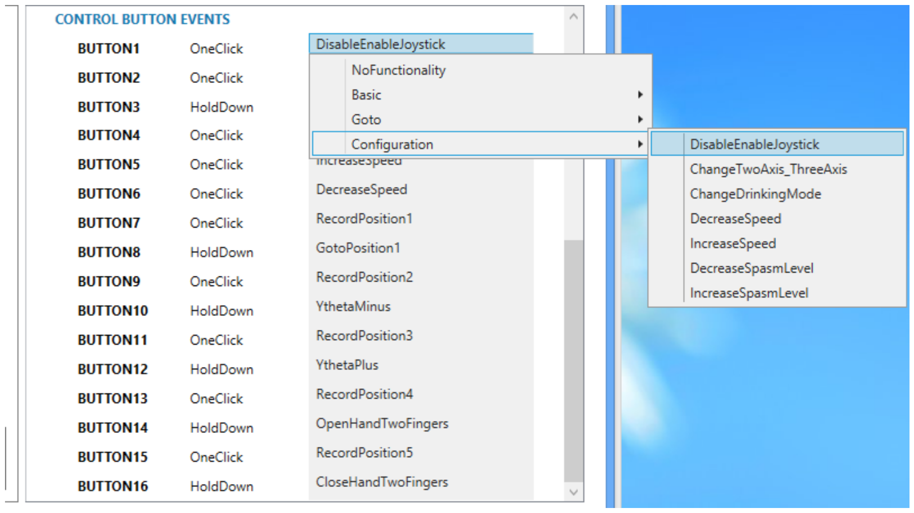


CONTROL STICK EVENTS

InclineLR	Minus	XMinus
	Plus	XPlus
InclineFB	Minus	YMinus
	Plus	YPlus
Rotation	Minus	ZMinus
	Plus	ZPlus
MoveLR	Minus	OpenHandThreeFingers
	Plus	CloseHandThreeFingers
MoveFB	Minus	XthetaMinus
	Plus	XthetaPlus
PushPull	Minus	ZthetaMinus
	Plus	ZthetaPlus

Select an Associated functionality/Configuration for each Control Button Event (Command/State) (see Figure 43).

Figure 43. Associated functionality/Configuration.

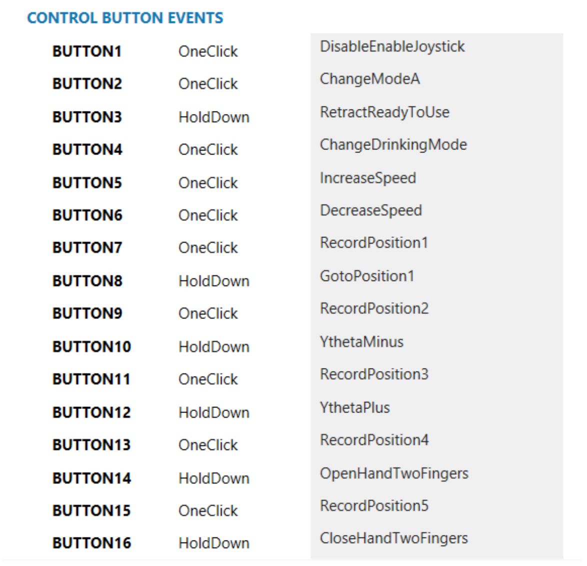


CONTROL BUTTON EVENTS

BUTTON1	OneClick	DisableEnableJoystick
BUTTON2	OneClick	
BUTTON3	HoldDown	
BUTTON4	OneClick	
BUTTON5	OneClick	IncreaseSpeed
BUTTON6	OneClick	DecreaseSpeed
BUTTON7	OneClick	RecordPosition1
BUTTON8	HoldDown	GotoPosition1
BUTTON9	OneClick	RecordPosition2
BUTTON10	HoldDown	YthetaMinus
BUTTON11	OneClick	RecordPosition3
BUTTON12	HoldDown	YthetaPlus
BUTTON13	OneClick	RecordPosition4
BUTTON14	HoldDown	OpenHandTwoFingers
BUTTON15	OneClick	RecordPosition5
BUTTON16	HoldDown	CloseHandTwoFingers

Figure 44 shows the Associated functionalities/Configuration selected for this lab setup.

Figure 44. Control button events.

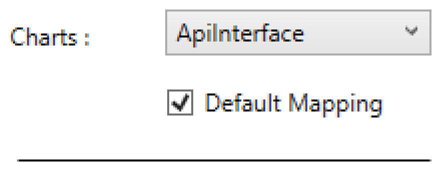


CONTROL BUTTON EVENTS

BUTTON1	OneClick	DisableEnableJoystick
BUTTON2	OneClick	ChangeModeA
BUTTON3	HoldDown	RetractReadyToUse
BUTTON4	OneClick	ChangeDrinkingMode
BUTTON5	OneClick	IncreaseSpeed
BUTTON6	OneClick	DecreaseSpeed
BUTTON7	OneClick	RecordPosition1
BUTTON8	HoldDown	GotoPosition1
BUTTON9	OneClick	RecordPosition2
BUTTON10	HoldDown	YthetaMinus
BUTTON11	OneClick	RecordPosition3
BUTTON12	HoldDown	YthetaPlus
BUTTON13	OneClick	RecordPosition4
BUTTON14	HoldDown	OpenHandTwoFingers
BUTTON15	OneClick	RecordPosition5
BUTTON16	HoldDown	CloseHandTwoFingers

Click on Default Mapping box. (see Figure 45).

Figure 45. Default mapping box.



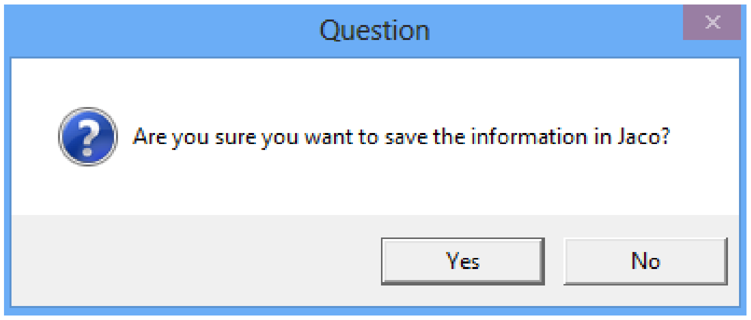
Charts : ApiInterface

☑ Default Mapping

Click on SEND button to save the configuration and information.

Click on YES in the Question dialog box to save the information in JACO (see Figure 46).

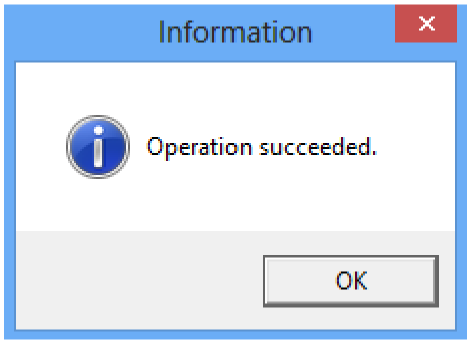
Figure 46. Question dialog box.


Question

Are you sure you want to save the information in Jaco?

Click OK in Information dialog box for Operation succeeded (see Figure 47).

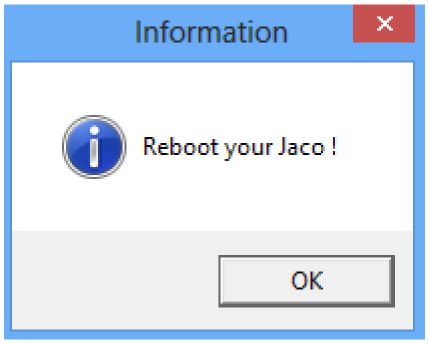
Figure 47. Information dialog box.


Information

Operation succeeded.

Click OK information box for Reboot your JACO (see Figure 48).

Figure 48. Information dialog box.


Information

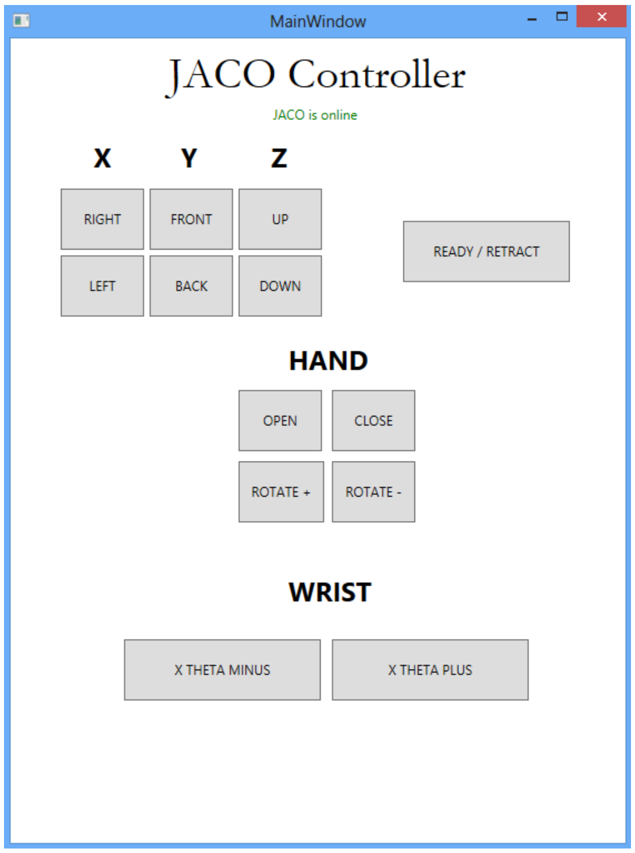
Reboot your Jaco !

Close the Mapping window.

Close the JACOSOFT software.

Open the JACO Controller project solution (see Appendix C) in Visual Studio/C#. Build it and Run it. Figure 49 will show the JACO Controller GUI.

Figure 49. JACO Controller GUI.



This lab setup allows the user to grasp and move objects using the JACO Controller GUI. The following figures show the manipulation of objects printed in a 3D printer by the JACO Controller GUI (see Figures 50 to 53).

Figure 50. JACO manipulating a cylinder.

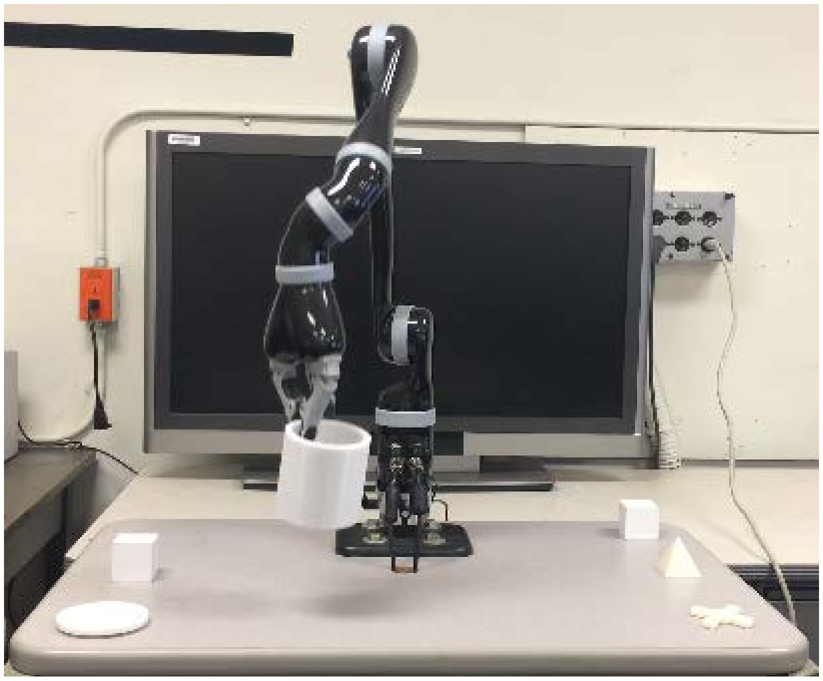


Figure 51. JACO arm manipulating a cube.

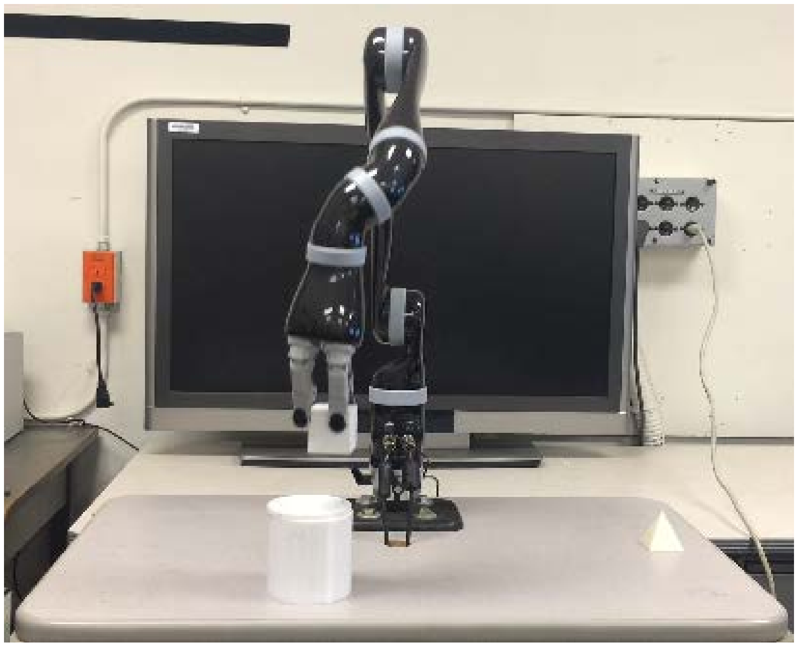


Figure 52. JACO arm manipulating a wheg.

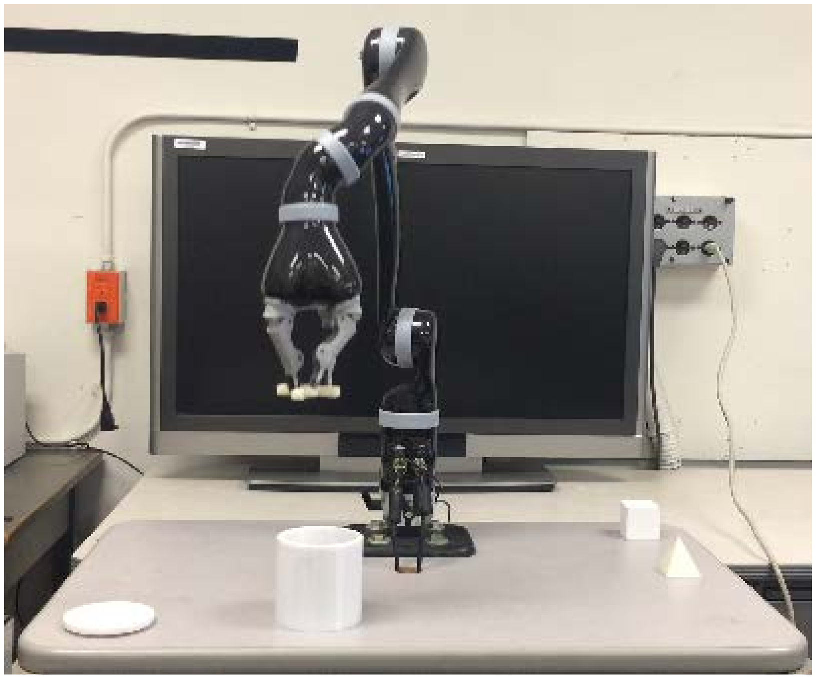
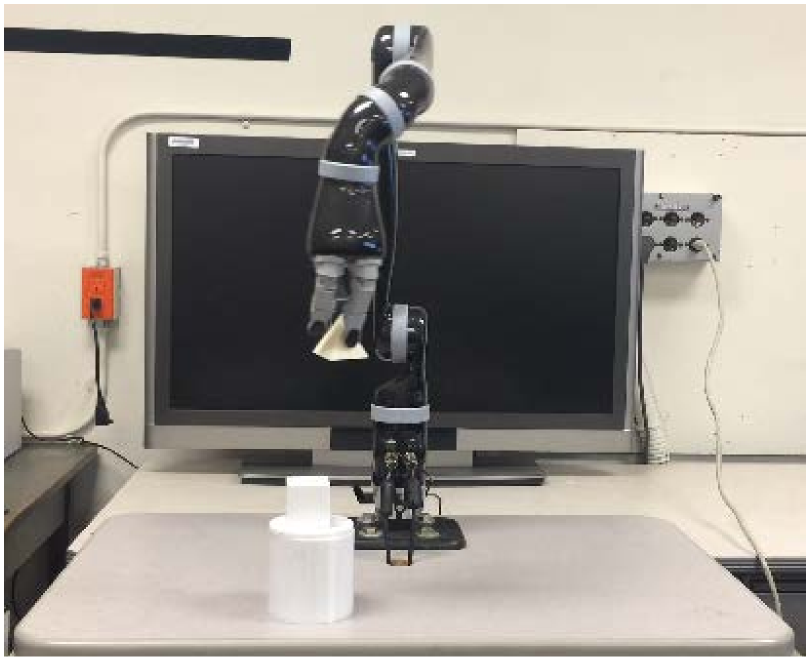


Figure 53. JACO arm manipulating a pyramid.



APPENDIX C

Based on Kinova C# [11] open source, were added the buttons required in order to have a GUI that allows the user to grasp and move objects from specific positions to required positions by a Virtual Joystick.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Threading;
using Kinova.DLL.Data.JACO;
using Kinova.API.JACO;
using Kinova.DLL.SafeGate;

namespace KinovaExample_Control
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        //The thread that will send button functionnality
        Thread m_WorkThread;

        //Delegate method for the thread
        private delegate void DoWorkDelegate();
        ManualResetEvent mre = new ManualResetEvent(false);

        //The password provided by Kinova
        private const string m_APIpass = "C6H12O6h2so4";

        //Your JACO
        private CJACOArm JACO;

        //Constructor of the window
        public MainWindow()
        {
            //Window xaml component init
            InitializeComponent();

            try
            {
                //Initialize JACO Object with an encrypted valid password
                JACO = new CJACOArm(Crypto.GetInstance().Encrypt(m_APIpass));


                //Take control of JACO
                JACO.ControlManager.StartControlAPI();
            }
            catch (Exception ex)
            {
                System.Console.WriteLine("Cannot connect to JACO");
            }

            //Check if JACO is connected
            if(JACO.JACOIsReady())
            {
                ((Label)m_Grid.Children[12]).Foreground = Brushes.Green;
                ((Label)m_Grid.Children[12]).Content = "JACO is online";
            }

            //Start the thread that will send joystick data to JACO
            m_WorkThread = new Thread(new ThreadStart(DoWork));
            m_WorkThread.Priority = ThreadPriority.BelowNormal;
            m_WorkThread.IsBackground = true;
            m_WorkThread.Start();
            mre.Set();
        }

        //Closing the window EVENT
        private void Window_Closing(object sender,
        System.ComponentModel.CancelEventArgs e)
        {
            m_WorkThread.Abort();
            JACO.ControlManager.StopControlAPI();
        }

        //Thread's work
        [STAThread]
        private void DoWork()
        {

            System.Windows.Threading.DispatcherOperation stdo = null;
            DoWorkDelegate updateJoystickDelegate = new
            DoWorkDelegate(UpdateJoystick);

            mre.WaitOne();


            while (m_WorkThread != null && m_WorkThread.IsAlive)
            {
                //Execute the UpdateJoystick method
                stdo = this.Dispatcher.BeginInvoke(System.Windows.Threading.
                DispatcherPriority.Normal, updateJoystickDelegate);
                Thread.Sleep(5);
            }
            if (stdo != null)
            {
                stdo.Dispatcher.InvokeShutdown();
        }
    }
    private void UpdateJoystick()
    {


        //Declare and initialize a CJoystickValue that will emulate a
        GeneralJoystick
        CJoystickValue value = new CJoystickValue();

        // X UP
        if (((Button)m_Grid.Children[4]).IsPressed)
        {
            value.InclineLR = -1f;
        }

        // Y UP
        if (((Button)m_Grid.Children[5]).IsPressed)
        {
            value.InclineFB = -1f;
        }


        // Z UP
        if (((Button)m_Grid.Children[6]).IsPressed)
        {
            value.Rotate = 1f;
        }


        // X DOWN
        if (((Button)m_Grid.Children[7]).IsPressed)
        {
            value.InclineLR = 1f;
        }


        // Y DOWN
        if (((Button)m_Grid.Children[8]).IsPressed)
        {
            value.InclineFB = 1f;
        }


        // Z DOWN
        if (((Button)m_Grid.Children[9]).IsPressed)
        {
            value.Rotate = -1f;
        }

        // Store Position
        if (((Button)m_Grid.Children[10]).IsPressed)
        {
            value.ButtonValue[2] = 1;
        }

        // Go To Position
```

```
            if (((Button)m_Grid.Children[11]).IsPressed)
            {
                value.ButtonValue[7] = 1;
            }


            // OPEN
            if (((Button)m_Grid.Children[13]).IsPressed)
            {
                value.MoveLR = 1f;
            }

            // CLOSE
            if (((Button)m_Grid.Children[14]).IsPressed)
            {
                value.MoveLR = -1f;
            }

            // ROTATE CLOCKWISE
            if (((Button)m_Grid.Children[15]).IsPressed)
            {
                value.PushPull = 1f;
            }

            // ROTATE COUNTERCLOCKWISE
            if (((Button)m_Grid.Children[18]).IsPressed)
            {
                value.PushPull = -1f;
            }

            // XTHETAMINUS
            if (((Button)m_Grid.Children[19]).IsPressed)
            {
                value.MoveFB = 1f;
            }

            // XTHETAPLUS
            if (((Button)m_Grid.Children[20]).IsPressed)
            {
                value.MoveFB = -1f;
            }

            //We update JACO with the new data.
            JACO.ControlManager.SendJoystickFunctionality(value);
        }

        //If we click the Record button
        private void button1_Click(object sender, RoutedEventArgs e)
        private void OPEN_Click(object sender, RoutedEventArgs e)
        {
        }
        private void ROTATE_Click(object sender, RoutedEventArgs e)
        {

        }
    }
}
```

LIST OF REFERENCES

[1] A. Jacinto, “Unmanned systems: A lab-based robotic arm for grasping,” M.S. thesis, Dept. Physics, Naval Postgraduate School, Monterey, CA, 2015.

[2] JACO *Research Edition User Guide*, V1.0.0. Kinova, Boisbriand, Canada, 2011–2012.

[3] Craig, John J. *Introduction to Robotics Mechanics and Control*. 3rd ed. Upper Saddle River: Pearson Prentice Hall, 2005.

[4] R. D. Klafter, T. A. Chmielewski and M. Negin, *Robotic Engineering: An Integrated Approach*, 1st ed. Englewood Cliffs, NJ: Prentice Hall, 1989.

[5] JACO *DH Parameters of JACO R&D*, V1.1.5, Kinova, Boisbriand, Canada, 2013.

[6] Saeed B. Niku, *Introduction to Robotics, Analysis, Control, Applications*. 2nd ed. Hoboken New Jersey: John Wiley & Sons, 2011.

[7] Long Wang, Joseph DelPetro et al. “Highly-underactuated robotic hand with force and joint angle sensors.”

[8] F. Boucher. (2015, Sep.). “From need to innovation,” *IEE Robotics & Automation Magazine*. [Online].ieeexplore.ieee.org

[9] J. Capille, “Kinematic and experimental evaluation of commercial wheelchair-Mounted robotic arms,” M.S. thesis, Dept. Mechanical Eng., University of South Florida, Tampa, FL, 2010.

[10] *JACOSOFT User Guide*, V1.4.2, Kinova, Boisbriand, Canada, 2011–2014.

[11] Kinova. Kinova USB. (n.d.). [Online]. Available: Kinova/Documentation/API/ Examples/Windows/CSharp/KinovaExemple_Control.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California